Bellcomm

955 L'Enfant Plaza North, S.W.
Washington, D. C. 20024

B71 07001

date: July 2, 1971

to: Distribution

from: J. E. Nahra, M. P. Odle

subject: A Dynamic Programming Computer Program
Case 105-4

ABSTRACT

The Dynamic Programming Concept for multi-stage decision processes is illustrated via a simple example. Based on this concept, a computer program was developed which can, in theory, solve any multi-stage decision process that can be put into the State Space Format. In practice, the program is limited as to the size of the problem it can handle.

Discrete, dynamic, optimization problems with a limited number of state variables, a large number of constraints, and many alternative strategies to be evaluated subject to the constraints, are good candidates for this program. An approach was taken in which constraints are used to substantially reduce the well known dimensionality problem associated with Dynamic Programming. The program starts at a point and generates all the optimal solutions which satisfy the specified constraints. One or more of the optimal solutions generated can then be extracted from the class of many optimal solutions for desired analysis and use. A realistic space program planning application is used to illustrate the feasibility and usefulness of the concept as well as the computer program.

(NASA-CR-121350) A DYNAMIC PROGRAMMING COMPUTER PROGRAM (Bellcomm, Inc.) 78 p

N79-73163

Unclas
00/61 12091

FF No.
(NASA CR OR TMX OR AD NUMBER) (CATEGORY)
[redacted]

RECEIVED
NASA STI FACILITY
AUG 1971

TABLE OF CONTENTS

Bellcomm
955 L'Enfant Plaza North, S.W.
Washington, D. C. 20024

B71 07001

date: July 2, 1971

to: Distribution

from: J. E. Nahra, M. P. Odle

subject: A Dynamic Programming Computer Program
Case 105-4

MEMORANDUM FOR FILE

1.0 Introduction

There are four basic elements for every decision:

1. Goal(s) or Objective(s)
2. Limitations or Constraints
3. Alternative Strategies, and
4. Evaluating Criterion(a)

The first element is to establish a goal or a set of objectives to be achieved. This, perhaps, is the most difficult part of a decision process and depends on the decision maker as much as it does on the situation or problem to be resolved. The second part is to identify the limitations or constraints within which the acts of a decision process must be carried out. This is perhaps the least subjective of the four elements and depends mainly on the problem. The third element is to enumerate all possible alternative strategies that satisfy the constraints and achieve the established objective(s). For complex problems, this is usually the most tedious and time consuming portion of the decision process and where the computer can be a useful tool. And the last element is to establish a criterion(a) by which the alternative strategies can be compared and evaluated so as to choose a "best" or "optimal" strategy.

Dynamic Programming is a well known mathematical technique in generating optimal strategies for multi-stage decision processes. A serious disadvantage of this method, however, is dimensionality, that is, large amounts of information must be stored in computer memory even for problems with relatively few dimensions (3 or 4). An approach is used to substantially reduce the dimensionality problem.

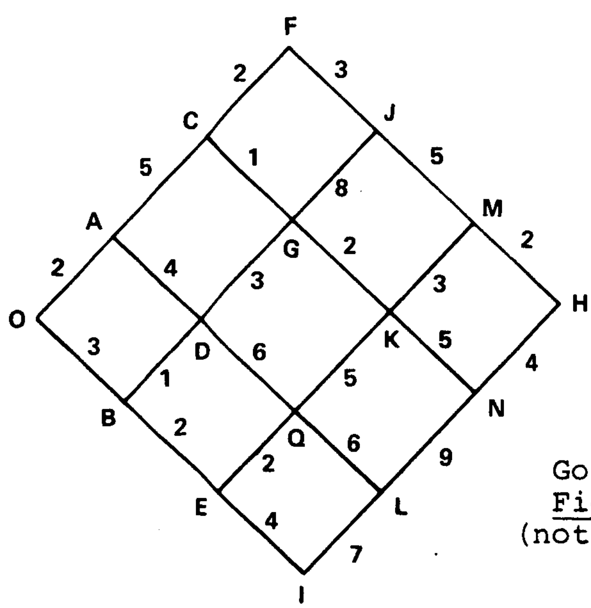
In Section 2, a simple example is employed to illustrate the concept of dynamic programming. The state space formalism is also explained and the procedure for the computer program is developed.

In Section 3, a description of the computer program is given. The steps a programmer must perform in order to set up his program for a particular application are specified in detail. The limitations of the dynamic programming package are also given.

In Section 4, a complex realistic decision process of a space program example is presented. The transformation of a qualitative engineering problem into the state space format, as well as the use of the program, are illustrated. Conclusions are then given in Section 5.

2.0 Dynamic Programming*

Perhaps the best way to explain the dynamic programming concept is through an illustrative example. Suppose you have just moved into a new house and you want to know the best route home from the office. You intuitively realize that you will be driving your car thru a limited number of paved streets connecting your office and your new home. You look at a city map and you chart all the possible ATTRACTIVE ROUTES connecting your office with your new home, and you record the distances in miles as shown in Figure 2-1. You then decide you want to take the shortest possible route.



Going Home
Figure 2-1
(not to scale)

*If the reader is familiar with the concept of Dynamic Programming, he may skip this section.

This is then a multi-stage decision process. You know where you are, that is at O (office) in Figure 2-1. You have decided on your objective or goal, that is, to get to H (your new home) in Figure 2-1. You have established your constraints most of which are intuitively obvious. First you must move along the lines (the streets) connecting your office and your house and you must always move to the right to get from your office to home. You also have decided on your evaluating criterion, that is, you want to choose the shortest distance route.

The only thing that remains is to enumerate and evaluate all the alternative paths that satisfy the constraints, and achieve your objective. Using the shortest distance criterion, you can choose the optimal path and your problem is solved!

2.1 Finding the Optimal Path

One possible way of finding the optimal path is to simply enumerate all 20 admissible paths that connect O with H, compute the distance of each and choose the one with the smallest value as your optimal solution. This is a reasonable approach and can easily be done for this problem. For more complex problems, however, this approach may not be feasible. Let's see if we can reduce the number of necessary calculations you have to make.

You are at O (office) in Figure 2-1, and you want to decide whether to go to vertex A or vertex B. These are the only two possible (admissible) paths that you can take. Suppose you knew the values of the shortest distance paths from A to H and from B to H. Then it is easy for you to decide whether to go to A or to B. You would add the value of the shortest distance path from A to H to the distance from O to A. Similarly, you would add the value of the shortest distance path from B to H to the distance from O to B. You would then compare the values of the two sums and choose the path that yields the smaller distance. So it is clear that you would have no trouble making the first decision and determining the overall value of the shortest distance path from O to H if you knew the values of the shortest distance paths from both A and B to H. Note that it is not the optimal path, but the value of the optimal path, that is the vital information.

Of course you don't know the values of the shortest distance paths between A, H and B, H. If you continue the same reasoning, however, you can easily find the shortest distance paths from both A and B to H if you knew beforehand the values of the best paths from C, D and E to H. You would continue this reasoning until you need only the values of the minimum-value paths from M and N to H in order to calculate the values of the shortest distance paths from J, K and L to H. But the shortest distances from M to H and from N to H are easily found since there is no free choice associated with picking an admissible path from either of these vertices to H. For each of these vertices, the value of the best and only admissible path is the distance between the vertex and the terminal point H.

Let us then put these ideas into practice. You start at H and then compute the minimum-path value associated with connecting vertex M with H which is 2; and with connecting vertex N with H, which is 4. You would then associate the numbers 2 and 4 with the vertices M and N respectively. You then would go back one more stage. At vertex J, you have no choice and must go to vertex M. You add the number associated with vertex M which is 2 to the distance from J to M which is 5 and you associate the number 7 with vertex J. At vertex K, you add the number associated with N which is 4 to the distance between K and M which is 5 to obtain 9. Since 5 is less than 9, you associate the value 5 with vertex K. For vertex L, there is only one admissible path L to N and the value associated with vertex L would then be 13. You continue this procedure until you reach your initial position, that is vertex O. The results are shown in Figure 2-2 where the value of the shortest distance path from each vertex to H is recorded.

All the required information to solve your problem is now available. You again would start at O (office) and ask the question whether to go to A or to B. This is now easy to answer since you know the value of the shortest distance paths from A and B to H. Because 11 plus 3 is less than 13 plus 2 you should proceed to B. Likewise from vertex B you proceed to D, G, K, M and finally H as shown in the figure. Note that you could have avoided making a decision as to which leg to proceed along at every vertex if you would have recorded the direction which initiates the shortest distance path from that vertex to H when you computed its optimal value on the backward sweep. The directions are denoted by arrows in Figure 2-3.

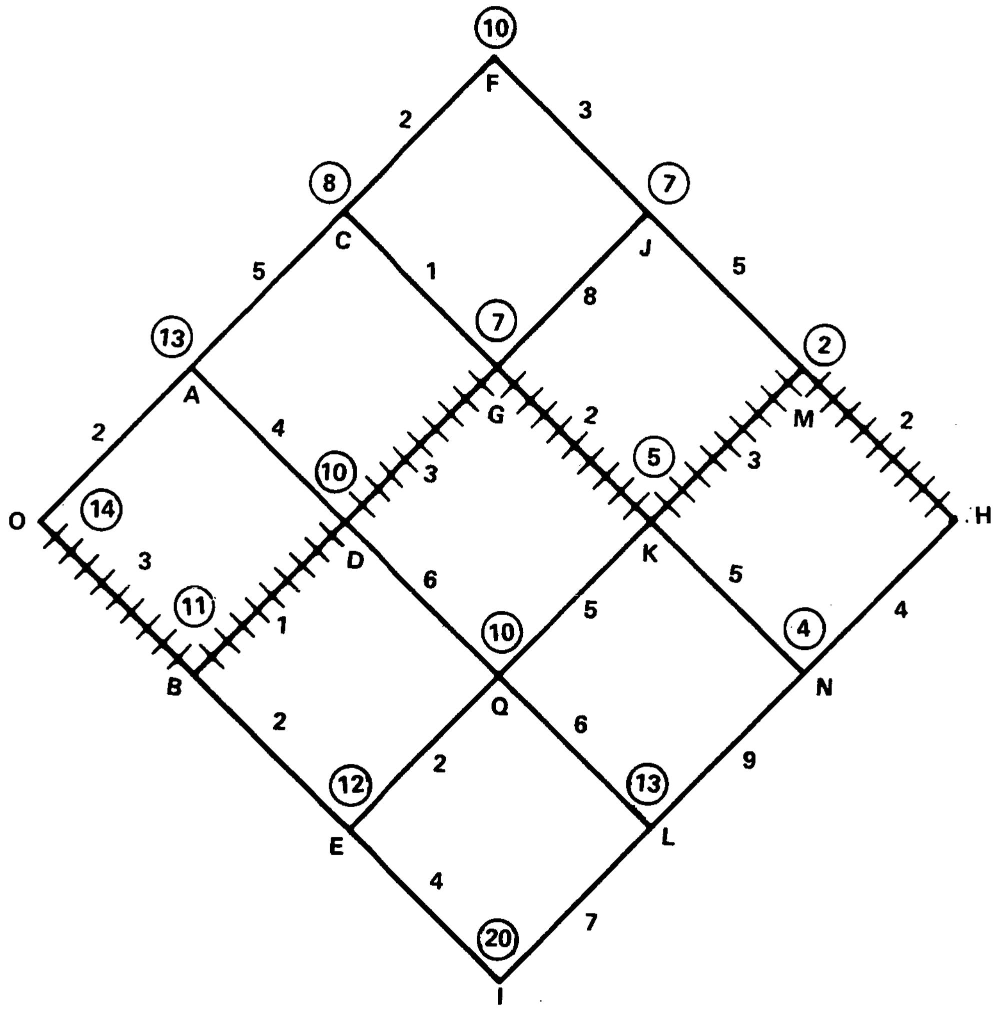
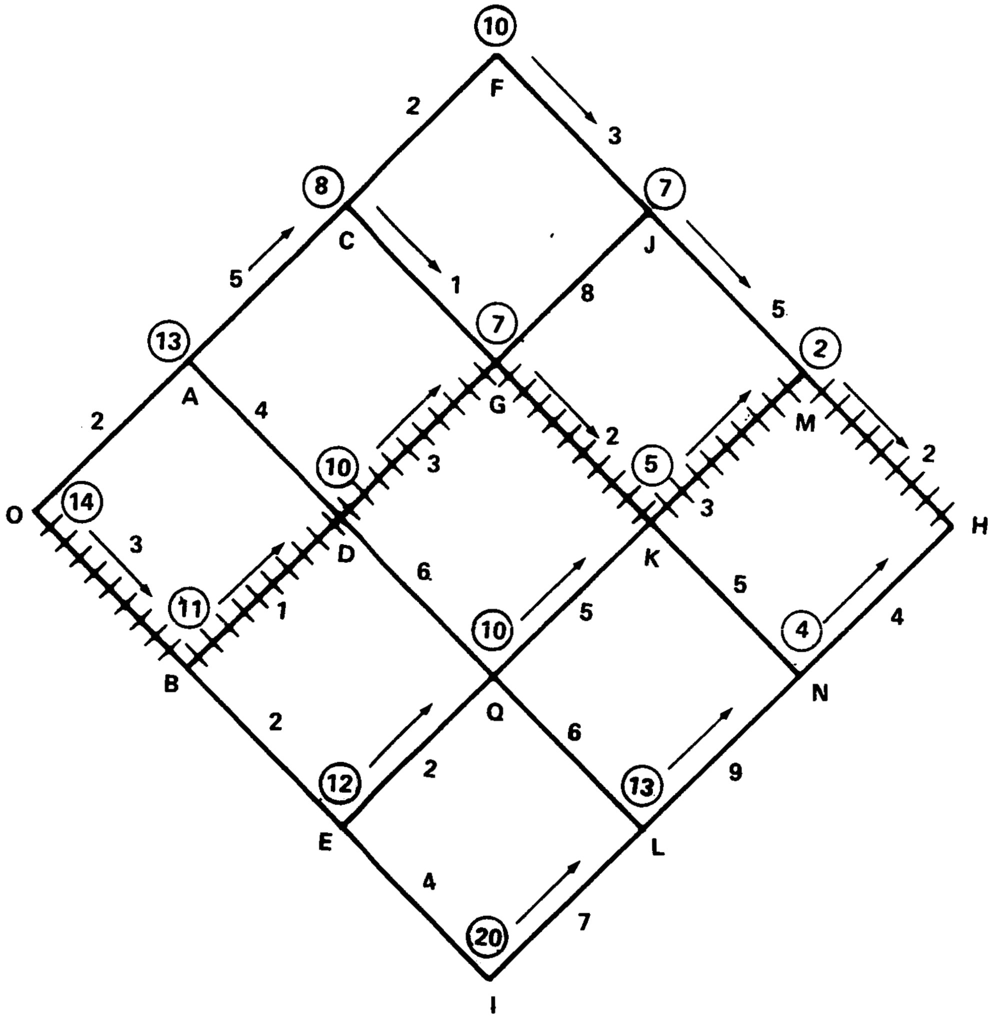

10
F
2
3
8
7
C
J
5
1
5
8
7
13
2
A
G
M
2
2
4
10
3
5
3
O
14
H
D
K
3
6
5
11
1
10
5
4
4
B
N
Q
2
6
12
13
9
2
E
L
4
20
7
I

Figure 2-2



Going Home
<u>Figure 2-3</u>

Your problem is thus solved. You start at O (your office) and you follow the arrows, that is, O to B, B to D, D to G, G to K, K to M and M to H. You, therefore, found the minimum distance route and its value between your office and your new home. If you look closely at Figure 2-3, you will find that you have much more information available to you than you have requested. You have not only solved the problem you want to solve, that is, the shortest distance route between O and H, but you have solved <u>all</u> the shortest distance route problems starting at <u>any</u> vertex in Figure 2-3 and terminating at vertex H! You have <u>imbedded</u> your <u>single</u> problem into a <u>class</u> of problems and found the solution to the <u>class</u> of problems.

This <u>extra</u> information might prove quite useful to you. To illustrate this fact, suppose one day you followed the minimum distance path from O to B. At B, however, you found that the street from B to D, which is along your minimum path, is <u>blocked</u> because of construction and you are forced to proceed to intersection E. You are now faced with the problem of finding the shortest distance path from E to H. If you had charted the minimum distance path from O to H only, you will need to compute the shortest distance route from E to H also. However, if you look at Figure 2-3, you will see that you already know the answer to your problem, that is, you follow the <u>arrows</u> from E to H. In fact, no matter what intersection point you <u>find</u> yourself <u>at</u> in the figure, you can easily find the shortest distance path from that point to your new home.

Of course the dynamic programming procedure could have been reversed, that is, you have been at your new home (H) and wanted to get to your office using the shortest distance route. Your initial point now becomes H and your final point O. The procedure is exactly the same.

2.2 <u>Efficiency of Dynamic Programming</u>

To evaluate the efficiency of the Dynamic Programming procedure, we can compare it with the direct enumeration method. For the dynamic programming approach, at each of the nine vertices where there was a real choice, two additions and one comparison were performed and at six other vertices, one addition was performed. For the direct evaluation approach, 20 admissible paths would have had to been enumerated, which would have involved five additions per path, yielding <u>100</u> additions, and a comparison of <u>20</u> results.

The general formulas for the n-stage case for this class of problems (n = 6 legs in our example) better illustrate the computational savings. The dynamic programming algorithm involves $\frac{n^2}{2} + n$ additions, while the direct enumeration generates

$$\frac{(n-1)\,n!}{(\frac{1}{2}n)!\ (\frac{1}{2}n)!}$$

additions. For n=20, dynamic programming requires an easily manageable 220 additions, while enumeration would require more than 1,000,000 additions.[1]

This is not really a good comparison since the dynamic programming algorithm solves a complete class of problems and provides much more useful information than the enumeration technique which only solves one problem with given starting and ending points. For any other starting point, the enumeration procedure must be repeated.

2.3 The State Space Formulation

In the above example we have intuitively introduced several important concepts. These concepts can be defined as follows: (the relationship of these concepts to the earlier example will be made shortly)

State - A state is the set of variables whose values describe the particular condition of the physical process that is being modeled. It is identified by a single number in some cases and by a set of numbers or a vector in others. (In the example, the state elements are the vertices in the figure.)

Stage - A stage is the position in the sequence of the particular decision process being considered. It is identified by a single number.

Control - A control is the set of decision variables that are under the control of the investigator. It is identified by a single number in some cases and by a set of numbers or a vector in others. In our example, it is the direction we choose to proceed.

<u>State Relations</u> - These are a set of relations that mathematically describe the outcome of a decision. They are usually a set of difference or differential equations. There are the same number of these relations as there are state variables.

<u>Cost Criterion</u> - This is the evaluating criterion that will determine the specific choice of control or decision variables. It must be a scaler and is identified by a single number.

<u>Constraints</u> - These are the limitations on our actions or our choice of control variables. There are different types of constraints as follows:

<u>State</u> - Restrictions on the admissible states.

<u>Control</u> - Restrictions on the allowable controls or decisions.

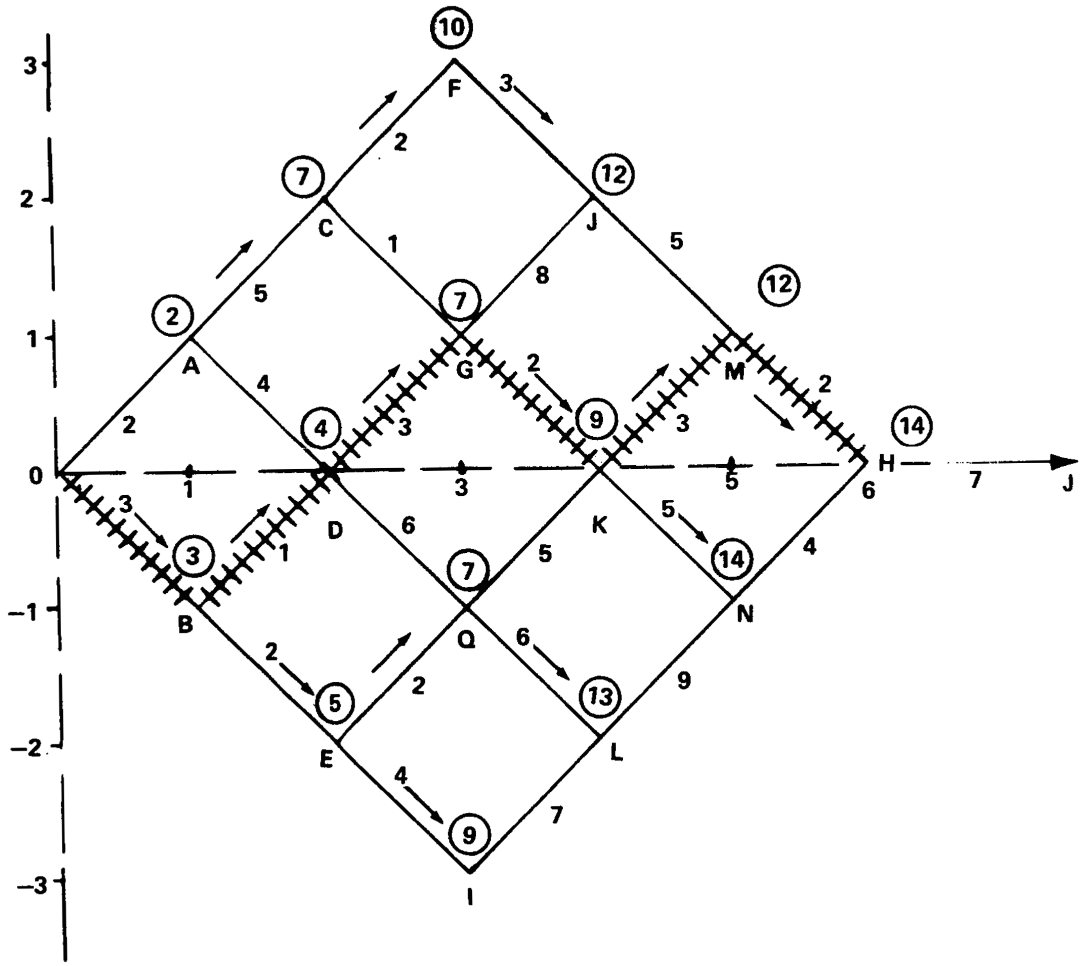
<u>Mixed</u> - Restrictions on the selection of both states and controls.

<u>Cost Function</u> - Limitations on the allowable costs.

All of these constraints can be expressed in terms of equations and/or inequalities.

Using this format, a decision process begins by first choosing the state, stage and control variable(s); the initial value (where you are); the final state value or stage value; the state relations; the cost criterion; the constraints, and finally finding the optimizing sequence of controls or decision variables. The best way to understand this procedure is to follow an example. Let us see if we can put the previous example in this format.

Let X be the state variable, J the stage variable, and U the control variable. Figure 2-3 can then be put in the form of Figure 2-4 where a coordinate system is now introduced. Note that the reverse problem is used, i.e., "going to the office" instead of "going home". The cost function (criterion) can be put in table form and depends on J, X and U. The vertices or intersections can now be identified by the coordinates, i.e., vertex C is the same as vertex (2,2) and likewise vertex M is (5,1). We chose the stage variable J to coincide with the abcissa so that only one state variable is needed. Note that J contains information describing the decision position in the sequence as well as the location. This may not always be possible and is done here for illustrative purposes.



Going to the Office
<u>Figure 2-4</u>

The cost function, therefore, is in the following form:

$$P(J,X,U) = \text{scalar} \qquad \text{where} \quad \begin{array}{l} J = 0 \to 6 \\ X = -3 \to +3 \\ U = \pm 1 \end{array}$$

For example,

$$P(3,-1,1) = 5$$

That is, when you are at intersection (3,-1) or Q in the figure and you apply a control +1 (to go to K), this decision will cost you 5 miles. Likewise,

$$P(5,1,-1) = 2.$$

The state equation in this case is simply the addition of the present state and the control to obtain the new state, i.e.,

$$X(J+1) = X(J) + U(J)$$

The state constraints are the limitations on the state variable X, i.e., X must lie within the boundaries you have chosen. Algebraically, these can be expressed as follows:

<u>State Constraints</u>

$$-J \leq X \leq J \text{ for } 0 \leq J \leq 3$$

$$J - 6 \leq X \leq -J + 6 \text{ for } 3 < J \leq 6$$

The control constraint is the limitation on your possible decisions. For this example:

<u>Control Constraint</u>

$$U = \pm 1$$

Note that this choice of control forces us to stay on the lines in the figure. There are no mixed or cost function constraints in this example.

The starting point is specified as

<u>Initial Conditions</u>

$$J = 0$$

$$X = 0$$

The final point is specified as:

Final Conditions

$$J = 6$$

$$X = 0$$

The problem is then to effect the transformation $(0,0) \rightarrow (6,0)$ with the control sequence which will satisfy the constraints and yield the shortest distance.

2.4 The Dynamic Programming Algorithm

As was shown earlier, the dynamic programming computation process begins by calculating the value of the shortest distance path from every feasible intersection point to the terminal point starting at the second last stage and proceeding backward until the initial stage is reached. With every feasible (admissible) state (intersection point), therefore, there corresponds one value of the optimal (shortest distance) path from that state to the terminal state. These values can be identifed by a table which is known as the Optimal Value Table or Function and is denoted by $V(J,X)$, e.g.,

$$V(2,2) = 7$$

as can be seen from Figure 2-4. Once the Optimal Value Function is computed, then all the optimal paths within the feasible region (satisfying the constraints) which terminate at the final point can easily be found, as was demonstrated earlier.

If the reverse problem was solved, i.e., you were at home and wanted to find the shortest distance path to your office; then you would start at O and proceed toward H in calculating the optimal value function V. The optimal values at the intersections will of course be different as can be seen by comparing figures 2-3 and 2-4. Note that the optimal solution in both cases is the same, as one would expect. The class of problems solved in this case, however, are all the feasible solutions that terminate at O. Or looking at it in another way, all the optimal solutions starting at O and terminating anywhere in the feasible region. This second procedure is used in the following algorithm:

Forward Sweep

1. Definitions:*

   X - an n dimensional vector
   U - an m dimensional vector, $m \leq n$
   J - stage or time counter
   P - a scalar function (or table) of X and U
   V - a scalar function (or table) of X and U

2. Let J = 0, X(J) = X_o, a specified initial vector and V = 0.

3. Search all possible permutations of U.

4. Test each permutation. If admissible, continue, if not go to next permutation.

5. For every admissible U calculate P = P(X,U).

6. Test P. If admissible continue, otherwise go to the next permutation.

7. For every admissible U, find all admissible X(J+1)'s from X(J+1) = X(J) + U(J).

8. At J+1, X(J+1) calculate V = V[J, X(J)] + P.

9. Compare V with the previously stored value of V at J+1, X(J+1), and save the smaller of the two as V[J+1, X(J+1)]; also save the corresponding vector U.

10. Go to the next stage; i.e., let J = J+1.

11. Repeat the procedure from step 3 through 10 for each reachable point X defined in 7.

12. At the final stage (J), print J, X and V(J,X) for all reachable values of X.

The forward sweep determines the Optimal Value Function (J,X), i.e., the optimal value table in our previous example. This table contains the optimal solutions

*See Section 2.3 for a complete definition of these quantities.

for the problems with the same initial point and constraints and varying final conditions. To obtain a particular solution, the following "backward sweep" is performed:

<u>Backward Sweep</u>

1. At $J = J_{final}$ let X = XF a specified vector.

2. Print J, X, U and V where U and V had been stored.

3. Let X = X-U

4. Let J = J-1

5. Go back to 2 and repeat until the first stage is reached.

X and U were generalized to include n and m number of variables respectively. A flow chart of this algorithm can be found in Reference 2.

Several important characteristics of this algorithm should be noted. The Optimal Value Function (Table) has to be stored as it is generated and must include every feasible state point. The number of state points increases geometrically with the number of state variables (dimensions). For the simple example presented earlier, there were 16 feasible state points for one variable x; for two and three variables there would have been 64 and 256 points respectively. If we let α denote the number of levels each state variable is allowed to take, n the number of state variables, and J the number of stages, then the number of possible state points would be $\alpha^n J$. For n=10, α=10 and J=20, which is a reasonable size problem, the number of state points would be 2×10^{11}, obviously a much larger number than any present day computer can store. Dimensionality, therefore, is the basic problem in dynamic programming applications.

In observing a human being execute a decision, it is seen that a relatively small number of <u>reasonable</u> alternatives are <u>considered</u>, although a large number of possible alternatives exist. The same concept can be used here. In order to solve a fairly complex problem using dynamic programming, the various constraints must be formulated so that only <u>reasonable</u> alternatives are considered by the program.

One important characteristic of this algorithm is the fact that computation of the optimal value function is started at the initial state which is usually known and is propagated forward always within the feasible region.

Another tacit assumption that was made in developing the dynamic programming algorithm is that the cost criterion function at a particular stage depends only on the state and control of that stage. It is not affected by information of previous or future stages.

In the following section the computer program implementing this algorithm is described. In Section 4 a realistic application is presented illustrating the use of the program and demonstrating the feasibility of this approach.

3.0 Computer Program Description

A computer program, GPALG, has been written which implements the previously described algorithm. The implementation provides a capability for the user to particularize the program to his problem. The user provides the constraints and functions which define his problem via FORTRAN statments in a PDP element. Upon compilation, skeletal subprograms on the Fastrand file are expanded to include the FORTRAN statements in the PDP element. After defining the class of problem using the PDP element, the program accepts inputs for a specific execution via two NAMELISTS. In general, the user must do the following steps to run the program:

1. Assign his own Fastrand program file (previously catalogued).

   ```
   @ASG,AX              USER*USERFILE.
   ```

2. Copy the symbolics and relocatables of the program from file GPDALG*GPDFIL.

   ```
   @COPY,SR             GPDALG*GPDFIL.,USER*USERFILE.
   ```

3. Enter his own problem definition via a PDP element.

   ```
   @PDP,FLIX            USER*USERFILE.ELEM,.ELEM
   ```

4. Recompile onto his file the symbolic elements with names NEXTST, COST, COSLIM, COMCOS, CONSTR, FINVEC. Ex:

```
@FOR,S              USER*USERFILE.COST,.COST
```

5. Pack and prep his file:

```
@PACK               USER*USERFILE.

@PREP               USER*USERFILE.
```

6. Map to create an absolute element for execution.

```
@MAP,IS             USER*USERMAP,.USERMAP

     LIB            USER*USERFILE.

     IN             USER*USERFILE.GPALG
```

The above steps serve to transfer the general program to the user and define his specific application. If the problem definition, as specified by the PDP element, is satisfactory, these operations are done only once and the program is now ready for execution. Input for program execution must also be provided and is described in the section "Problem Input". A complete listing of the program is given in the Appendix, and is available on Fastrand File GPDALG*GPDFIL.

3.1 The PDP Element

For initial problem definition, a PDP element (see subsection 4.3 for example) with various entry points must be provided by the user.[3] Between each entry point name and its corresponding END, the user must provide FORTRAN statements which either define a function (like the "next state" function) or define and test constraints (like the "state" constraints). The specific FORTRAN statements provided depend on the function of that entry point. Table 3-1 shows entry point names, function and FORTRAN variables to the user in his FORTRAN statements.

Name	Function	Variables Available
NEWST	next state function	CS,CP,ST,NX,NC,JTIME,XIN
COSTF	penalty or cost function	CS,CP,NX,CC,JTIME,XIN,P
TCOST	total cost or value function	CS,CP,ST,P,V,VV,JTIME,XIN
VCONST	total cost constraints	CS,CP,ST,P,V,VV,JTIME,XIN
SCONST	state constraints	CS,CP,NX,NC,JTIME,XIN
CCONST	control constraints	CS,CP,NX,NC,JTIME,XIN
BCONST	mixed (state & control constraints)	CS,CP,NX,NC,JTIME,XIN
COSTCN	penalty constraints	P
FINCNS	final constraints	CS,JTIME,NX,XIN

Table 3-1

Table 3-2 defines the variables shown on the right in Table 3-1.

Name	Type	Definition
CS	Int (Integer)	current state vector
CP	Int	current control vector
NX	Int	number of elements in CS,XIN, and ST
NC	Int	number of elements in CP
ST	Int	new state vector (generated by CS and CP)
P	Real	penalty cost function
V	Real	Current accumulated optimal value function
VV	Real	total accumulated optimal value function including going from CS to ST (VV=V+P)
JTIME	Int	current time or stage
XIN	Int	initial state vector

Table 3-2

Referring to Table 3-1

In entry point NEWST the next state vector computed must be placed in variable ST.

In entry point COSTF, the penalty or cost should be placed in variable P.

In entry point TCOST, the total accumulated cost should be placed in variable VV.

For entry points VCONST-FINCNS where constraints are being tested, the FORTRAN IF statements must have a GO TO 1000 if a constraint is not satisfied.

3.2 Changing the Problem Definition

If after the problem is defined, the user wishes to modify that definition by changing one or more FORTRAN statements in the PDP element, some subroutines will need to be recompiled. Table 3-3 shows which subroutine is recompiled if statement(s) in an entry point are changed:

ENTRY POINT	SUBROUTINE
NEWST	NEXTST
COSTF	COST
COSTCN	COSLIM
TCOST or VCONST	COMCOS
SCONST CONST, or BCONST	CONSTR
FINCNS	FINVEC

Table 3-3

Each time that a recompilation is done, the EXEC 8 FurPur operations of PACK, PREP and MAP must also be done so as to create a new absolute element of the program for execution (see Section 3.0).

When the user is satisfied with the problem definition, the program is ready to be executed.

3.3 Problem Input and Output

For a specific running of the program, input and output are controlled via two FORTRAN namelists (CONDAT and PRTDAT). Most of the variables in namelist CONDAT are for problem definition. Variable IØUT controls program termination and output. The result of execution of the program is an array of feasible states and the associated accumulated cost of getting from the initial state to those states. The user may specify one of two types of terminal conditions. The first is to specify a final stage number so that the programs runs through n stages where n is the difference between the final and initial number of stages plus one. The second is to specify a final set of constraints. In this case the program will move from stage to stage until the final constraints are satisfied. The feasible state vectors occurring at this final stage are then the output of the program and all feasible states at every stage are stored on a file if the user desires. Associated with each feasible final state is a state number which is used for identifying the final state for which the backward problem is to be worked. By backchaining (backward sweep) is meant the process of tracking from a specific final state to the initial state. The path provided by the program is optimal (least cost). The feasible final states to be used in backchaining are specified in namelist PRTDAT.

1. NAMELIST/CONDAT/

Name	Type/array Size	Definition
JIN	Int	initial stage
XIN	Int/20	initial state vector
VINIT	Real	initial cost
NX	Int	# of components in state vector (≤ 20)
NCONTR	Int	# of components in control vector (≤ 20).
U	Int/20x11	U(i,1)=# of values of the control's i^{th} component. U(i,j+1)=j^{th} value that the i^{th} component assumes.
MAXJ	Int	last stage if stage used as limit
ID	Alphabetic	6 character problem identifier

Name	Type/array Size	Definition
IOUT	Int	controls termination of problem = 1 MAXJ specified and results stored on unit 10. Program terminates. = 2 Same as 1 except namelist PRTDAT requested for backchaining before termination = 3 Final constraints specified and results stored on unit 10. Program terminates = 4 Same as 3 except namelist PRTDAT requested for backchaining before termination = 5 Results from previous execution exist on file 10. Request PRTDAT for backchaining
NAMELIST/PRTDAT/ (used when IOUT = 2, 4 or 5)		
ID	Alphabetic	6 character problem identifier
XFIN	Int/1000	Vector of state numbers the user would like to see backchained (these numbers are given as previous output)
NFIN	Int	Length of XFIN vector ($\leq$1000)

If in NAMELIST/CONDAT/, IOUT=5 is specified, all computation is bypassed and only backchaining is done. The user must have, therefore, previously run the program with IOUT$\neq$5 and catalogued a FASTRAND file on which the results were to be saved. When the user wishes to save results for later use, he must have the following control cards:

```
@ASG,A          USER*USERDATA.

@USE            10,USER*USERDATA.
```

and again specify that data file as unit 10 when backchaining is done.

3.4 Notes on the Implementation

The algorithm is most useful when the number of feasible states between the initial stage and final stage is large. For many problems this number may be as large as 10 to 15 thousand. Because it is not possible to keep in main core memory all the feasible states generated for a problem of this size, a paging scheme for keeping feasible states on a mass storage device was implemented. The paging scheme is such that each feasible state is accessed by reference to its page number and relative position within the page. This scheme was possible because the algorithm only requires direct access to states at stage n for computation of states at stage n+1. The feasible states at stage 0,...,n-1 can thus be stored externally until they are needed for back-chaining. The subroutine PROFILE in the Appendix implements this paging scheme. Another technique to alleviate the core storage problem was a compact representation of the state and control vectors reducing the number of memory words needed for these vectors by a factor of 4.

3.5 Limitations of the Program

The state and control vectors are limited to 20 components each. Each component of the control vector may take on at most 10 values. The number of final states to be backchained (in namelist PRTDAT) is limited to 1000. The most important limitation of the program involves the number of feasible states generated at any given stage. Because all feasible states at a stage n are necessary to generate feasible states at stage n+1, both stages must be able to fit into main memory simultaneously. The limiting size on the number of feasible states at time n and n+1 is 2500. If while generating feasible states at stage n+1 (from states at stage n) the sum is greater than 2500, the program will terminate. Since the number of feasible states generated depends on the problem constraints, tightening the constraints may allow the problem to be completed if this maximum is exceeded.

4.0 Space Program Example

In developing a space program plan, a critical problem is usually scheduling the development of major program segments in the best way possible within cost and other constraints. Suppose several space program plans call for the continuation, initiation, and completion of six major program segments. These might be Apollo, Skylab I, Skylab II, Skylab III, Earth to Orbit Shuttle (shuttle), and Intermediate Launch Vehicle (ILV). Immediately, several important questions arise concerning this plan. First is it feasible within the time and cost constraints? If it is, what is the best schedule?

A typical set of cost data for each of these program segments is presented in Table 4-1. The time intervals are in years and represent the stage of normal development or continuation for a particular program segment. The costs are in millions of dollars. The arrow identifies a key level in the program which might denote the first launch.

YEARS →	1	2	3	4	5	6	7	8
SEGMENT								
1 APOLLO	653	287	↓85					
2 SKYLAB I	372	474	↓201	6				
3 SKYLAB II	37	80	218	↓353	588	49		
4 SKYLAB III	45	135	225	↓300	130			
5 SHUTTLE	49	200	550	600	↓450	450	200	
6 ILV	19	57	105	192	↓230	230		
FIXED COSTS	363	415	386	354	347	349	350	350

Table 4-1

The following is a typical set of qualitative constraints on the program plan:

1. Yearly expenditures must not exceed \$1.525 billion the first five years and \$2 billion thereafter.

2. Yearly expenditures must not be less than \$1.0 billion for the first three years.

3. Maximum time allowed for the plan is ten years.

4. Apollo has first priority and must be continued.

5. Skylabs must follow in order and it is desirable to have them at least one year apart.

6. It is desirable to have the ILV (Intermediate Launch Vehicle) at the same time or before the Shuttle.

7. It is desirable to have the first shuttle launch by 1978.

8. It is also desirable to assure a program's progress once initiated.

4.1 Brute Force Approach

This problem can be posed as a decision process. The goal is to complete the specified program segments. The constraints are listed above. The evaluating criterion is to minimize overall cost; and the alternative strategies are quite numerous. If there were only the time constraint, and assuming the first program segment (Apollo) is essentially fixed, then one can show that more than 8×10^7 combinations of segments that would yield a completed program in 10 years are possible. One can then compute the cost for each of these programs, and choose the one that yields the smallest value. This is of course beyond the capability of a human being.

A more reasonable approach would be to take advantage of the constraints and eliminate many of the possibilities. Since the first segment (Apollo) is essentially fixed, we see from the cost table that the expenditures for the first year for Apollo and fixed costs, is approximately 1.0 billion dollars. We have then approximately 1/2 billion dollars to initiate new programs and stay within the expenditure constraint. From Constraint 7, we notice that the shuttle has to reach level 5 by 1978. We also notice that the Skylabs must be initiated in order and at least one year apart. Also that the ILV development must precede or correspond with shuttle development. We must be careful not to start too many programs because their peak expenditures might occur at the same time and the yearly expenditure constraint might be violated at a later year. And to complicate the situation even further, the lower bound on yearly expenditures, that is, 1.0 billion dollars might be violated. An experienced person would probably be able to formulate a feasible program, that satisfies the constraints, within a reasonable time. However, there is no assurance that the program he formulated is the best one possible in the sense of overall minimum cost!

4.2 The Space Program Example in State Space Format

Following the procedure outlined in Subsection 2.3, we can now formulate the space program example in state space format.

State Variables - The space program segments constitute the physical system under consideration, therefore, define six state variables corresponding to the six program segments, i.e.,

$X_1 \equiv$ Apollo

$X_2 \equiv$ Skylab I

$X_3 \equiv$ Skylab II

$X_4 \equiv$ Skylab III

$X_5 \equiv$ Shuttle

$X_6 \equiv$ ILV

The numerical values or the levels of these variables describe the state of the system.

Stage Variable - Time describes the position in the sequence of the decision process in this example. We, therefore, define the stage variable as time, i.e.,

$$J \equiv t \text{ (years)}$$

Any other time period can of course be used.

Control Variables - These are the variables under our control which affect the state of the system. In this case these are decisions on the development of the various program segments. Each program segment has a corresponding decision variable. Let us denote these as follows:

U_1 - Development decisions on Apollo (X_1)

U_2 - Development decisions on Skylab I (X_2)

U_3 - Development decisions on Skylab II (X_3)

U_4 - Development decisions on Skylab III (X_4)

U_5 - Development decisions on Shuttle (X_5)

U_6 - Development decisions on ILV (X_6)

These variables may take on values of 0, 1 and 2 where 0 would denote no development or a delay in a program segment, 1 would denote normal development in the segment, and 2 would denote accelerated development in the program segment so that two years of normal program development can be accomplished in one year of actual time. Of course, more levels can be used if desired.

State Equations - The state equations mathematically describe the outcomes of decisions at every stage in the process. In this case, these are very simple linear difference equations as follows:

$$X_1(J+1) = X_1(J) + U_1(J)$$

$$X_2(J+1) = X_2(J) + U_2(J)$$

$$X_3(J+1) = X_3(J) + U_3(J)$$

$$X_4(J+1) = X_4(J) + U_4(J)$$

$$X_5(J+1) = X_5(J) + U_5(J)$$

$$X_6(J+1) = X_6(J) + U_6(J)$$

The level of each of the state variables is changed by adding to it the value of its corresponding control (decision) variable which may be 0, 1 or 2.

Cost Function - The cost function (table) gives the cost incurred as a result of the particular state of the system and the particular decisions (control) taken. In this case, the cost is the development cost presented in Table 4-1. This table was modified to account for accelerating or delaying a particular program segment. Factors of 0.75 and 1.25 were used for delaying and accelerating a program respectively. If a program is delayed, the cost of keeping it at its current level would be 0.75 of the yearly expenditures at that level. If a program is accelerated, i.e., two years of development in one calendar year, the normal cost for the two years is increased by 25%. For example, the normal development for Skylab III the first 2 years would cost 45 and 135 million dollars respectively. To accomplish both years development level in one year would cost 1.25 times (45 + 135) or 225 million dollars, as shown in Table 4-2 where the cost data is presented.

X_1 which denotes the Apollo segment is initiated at level 10. This signifies that the segment is in its 10th year of development. The arrow points to the key level in a program segment as before. Fixed costs are denoted by V_o in the last row. Note that the cost penalty depends on the state, the stage, and the control. Once a program reaches its maximum level (completed), its corresponding control and cost are set to zero and infinity respectively.

Constraints - As before, the constraints can be grouped into four different categories: state, control, mixed and cost constraints. The qualitative constraints enumerated earlier can be put in equation or inequality form as follows:

State Constraints: Table 4-3 numerically describe some of the qualitative constraints mentioned earlier.

SEGMENT									
APOLLO	X_1 \ U_1	10	11	12	13				
	0	500	490	215					
	1	653	287	85	∞				
SKYLAB I	X_2 \ U_2	0	1	2	3	4			
	0	0	280	355	150				
	1	372	474	201	6	∞			
	2	1060	845	254	∞	∞			
SKYLAB II	X_3 \ U_3	0	1	2	3	4	5	6	
	0	0	28	60	164	265	440	37	
	1	37	80	218	353	588	49	∞	
	2	146	373	715	1180	800	∞	∞	
SKYLAB III	X_4 \ U_4	0	1	2	3	4	5		
	0	0	34	100	169	225	98		
	1	45	135	225	300	130	90		
	2	225	450	655	537	∞	∞		
ORBIT SHUTTLE	X_5 \ U_5	0	1	2	3	4	5	6	7
	0	0	37	150	412	450	338	338	150
	1	49	200	550	600	450	450	200	∞
	2	311	936	1440	1310	1125	937	∞	∞
INT. LAUNCH VEH. ILV	X_6 \ U_6	0	1	2	3	4	5	6	
	0	0	14	43	78	144	172	172	
	1	19	57	105	192	230	230	∞	
	2	95	202	370	527	575	∞	∞	
FIXED COST	Vo	363	415	386	354	347	349	350	→ CONSTANT

Table 4-2
Cost Function
In Millions of Dollars
$\alpha=.75,\ \beta=1.25$

	SEGMENT	KEY LEVEL	ACCEPTANCE KEY DATES MINIMUM	ACCEPTANCE KEY DATES MAXIMUM	MINIMUM FINAL STATUS
APOLLO	1	13	1973	1973	13
SKYLAB I	2	2	1973	1974	4
SKYLAB II	3	4	1974	1976	6
SKYLAB III	4	4	1976	1979	5
SHUTTLE	5	5	1976	1978	7
ILV	6	5	1976	1978	6

Table 4-3
Numerical Formulation of Qualitative State Constraints

where the key level again denotes the first launch date. Acceptance key dates are the key dates within which the key level must be achieved, and the minimum final status is the minimum level acceptable at the end of the designated time interval. For example, in Table 4-3, Skylab I which is segment 2 has a key level of three which means its first launch will occur after three years of normal development. This key level must occur either in 1973 or in 1974; and at the end of the program, the fourth year of development must be completed.

The limits on the state variables and the qualitative constraints can be translated into the following inequalities:

1. $10 \le X_1 \le 13$
2. $0 \le X_2 \le 4$
3. $0 \le X_3 \le 4$
4. $0 \le X_4 \le 5$
5. $0 \le X_5 \le 7$
6. $0 \le X_6 \le 6$
7. $X_2 < 2$ for $J < 3$
8. $X_2 \ge 2$ for $J \ge 4$
9. $X_2 \ge 4$ for $J = 10$
10. $X_3 < 4$ for $J < 4$
11. $X_3 \ge 4$ for $J \ge 6$
12. $X_3 \ge 6$ for $J = 10$
13. $X_4 < 4$ for $J < 6$
14. $X_4 \ge 4$ for $J \ge 9$
15. $X_4 \ge 5$ for $J = 10$
16. $X_5 < 5$ for $J < 6$
17. $X_5 \ge 5$ for $J \ge 8$
18. $X_5 \ge 6$ for $J = 10$
19. $X_6 < 5$ for $J < 5$
20. $X_6 \ge 5$ for $J \ge 8$
21. $X_6 \ge 6$ for $J = 10$
22. $X_6 \ge 6$ for $X_5 = 5$
23. $X_2 \ge 3$ for $X_3 = 4$
24. $X_3 > 4$ for $X_4 = 4$

The problem is then to find the control sequence that will transform the state of the system from its initial conditions to the desired final conditions satisfying the constraints and minimizing the overall cost of the program. The next subsection will describe the input to the dynamic programming computer algorithm and the resulting output.

4.3 Computer Program Input and Output Illustrated

Preparatory Input

The following operations prepare the program for execution. The statements in the PDP element define the constraints and functions for the space program problem.

```
@ASG,AX          GPDALG*STSPP.
@ASG,A           GPDALG*GPDFIL.
@COPY,SR         GPDALG*GPDFIL.,GPDALG*STSPP.
@PDP,FLIX        GPDALG*STSPP.DECS,.DECS
@FOR,S           GPDALG*STSPP.COST
@FOR,S           GPDALG*STSPP.COSLIM
@FOR,S           GPDALG*STSPP.COMCOS
@FOR,S           GPDALG*STSPP.CONSTR
@FOR,S           GPDALG*STSPP.FINVEC
@PACK            GPDALG*STSPP.
@PREP            GPDALG*STSPP.
@MAP,IS          GPDALG*STSP.MAP.,MAP
     LIB         GPDALG*STSPP.
     IN          GPDALG*STSPP.GPALG
```

The Fastrand file GPDALG*STSPP now has an absolute element on it which is ready for execution.

```
@PDP,FLIX           GPDALG*STSPP.DECS,.DECS
FINCNS              PROC
                    IF (CS(3) .NE. 6 .OR. CS(5) .NE. 7) GO TO 1000
 END
NEWST               PROC
                    DO 10  I=1,NX
                    ST(I)=CS(I)+CP(I)
 10                 CONTINUE
 END
COSTF               PROC
                    DIMENSION VO(10)
                    DIMENSION COSTAB(6,8,3)
                    DATA (VO(I),I=1,10)/363.,415.,386.,354.,347.,349.,350.,
      .             350.,350.,350./
                    DATA ((COSTAB(1,I,J),I=1,4),J=1,3)/
      .             500.,490.,215.,0.,653.,287.,85.,100000.,0.,0.,0.,0./
                    DATA  ((COSTAB (2,I,J),I=1,5),J=1,3)/
      .             0.,280.,355.,150.,4.,
      .             372.,474.,201.,6.,100000.,
      .             1060.,845.,254.,100000.,100000./
                    DATA ((COSTAB(3,I,J),I=1,7),J=1,3)/
      .             0.,28.,60.,164.,265.,440.,37.,
      .                37.,80.,218.,353.,588.,49.,100000.,
      .             146.,373.,715.,1180.,800.,100000.,100000./
                    DATA ((COSTAB(4,I,J),I=1,6),J=1,3)/
      .             0.,34.,100.,169.,225.,98.,
      .             45.,135.,225.,300.,130.,100000.,
      .             225.,450.,655.,537.,100000.,100000./
                    DATA ((COSTAB(5,I,J),I=1,8),J=1,3)/
      .             0.,37.,150.,412.,450.,338.,338.,150.,
      .             49.,200.,550.,600.,450.,450.,200.,100000.,
      .             311.,936.,1440.,1310.,1125.,937.,100000.,100000./
                    DATA ((COSTAB(6,I,J),I=1,7),J=1,3)/
      .             0.,14.,43.,78.,144.,172.,172.,
      .             19.,57.,105.,192.,230.,230.,100000.,
      .             95.,202.,370.,527.,575.,100000.,100000./
                    COST=0.
                    DO 10  I=1,NX
                    K=CP(I)+1
                    KK=CS(I)-XIN(I)+1
                    COST=COST+COSTAB(I,KK,K)
 10                 CONTINUE
                    COST=COST+VO(JTIME+1)
 END
COSTCN              PROC
                    IF (P .GT. 1525. .AND. JTIME .LT. 5)  GO TO 1000
                    IF (P .GT. 2000.)  GO TO 1000
                    IF (P .LT. 1000. .AND. JTIME .LT. 3)  GO TO 1000
 END
TCOST               PROC
                    VV=V+P
 END
VCONST              PROC
```

```
 END
 SCONST          PROC
                 NN=1
                 IF (CS(1) .LT. 10 .OR. CS(1) .GT. 13 )  GO TO 1000
                 NN=2
                 IF (CS(2) .LT.  0 .OR. CS(2) .GT.  4 )  GO TO 1000
                 NN=3
                 IF (CS(3) .LT.  0 .OR. CS(3) .GT.  6 )  GO TO 1000
                 NN=4
                 IF (CS(4) .LT.  0 .OR. CS(4) .GT.  5 )  GO TO 1000
                 NN=5
                 IF (CS(5) .LT.  0 .OR. CS(5) .GT.  7 )  GO TO 1000
                 NN=6
                 IF (CS(6) .LT.  0 .OR. CS(6) .GT.  6 )  GO TO 1000
                 NN=7
                 IF (JTIME .LE. 3 .AND. CS(2) .GT. 3)  GO TO 1000
                 NN=8
                 IF (JTIME .GT. 3 .AND. CS(2) .LT. 3)  GO TO 1000
                 NN=9
                 IF (JTIME .EQ. 10 .AND. CS(2) .LT. 4)  GO TO 1000
                 NN=10
                 IF (JTIME .LE. 4 .AND. CS(3) .GT. 4)  GO TO 1000
                 NN=11
                 IF (JTIME .GE. 6 .AND. CS(3) .LT. 4)  GO TO 1000
                 NN=12
                 IF (JTIME .EQ. 10 .AND. CS(3) .LT. 6)  GO TO 1000
                 NN=13
                 IF (JTIME .LE. 6 .AND. CS(4) .GT. 4)  GO TO 1000
                 NN=14
                 IF (JTIME .GE. 9 .AND. CS(4) .LT. 4)  GO TO 1000
                 NN=15
                 IF (JTIME .EQ. 10 .AND. CS(4) .LT. 5)  GO TO 1000
                 NN=16
                 IF (JTIME .LE. 6 .AND.(CS(5) .GT. 5  .OR. CS(6) .GT. 5) )
        .        GO TO 1000
                 NN=17
                 IF (JTIME .GE. 8 .AND.(CS(5) .LT. 5 .OR. CS(6) .LT. 5)  )
        .        GO TO 1000
                 NN=18
                 IF (JTIME .EQ. 10 .AND.(CS(5) .LT. 6 .OR. CS(6) .LT. 6) )
        .        GO TO 1000
                 NN=19
                 IF (CS(5) .EQ. 5 .AND. CS(6) .LT. 5)  GO TO 1000
                 NN=20
                 IF (CS(3) .EQ. 4 .AND. CS(2) .LT. 3)  GO TO 1000
                 NN=21
                 IF (CS(4) .EQ. 4 .AND. CS(3) .LT. 4)  GO TO 1000
 END
 CCONST          PROC
 END
 BCONST          PROC
                 NN=24
                 DO 10  I=2,NX
                 IF (CS(I) .GT. XIN(I) .AND. CP(I) .LE. 0 .AND. JTIME
        .        .LT. 5)  GO TO 1000
 10              CONTINUE
                 NN=25
                 IF (CS(1) .LT. 13 .AND. CP(1) .NE. 1)  GO TO 1000
 END
```

Problem Input

1. The following sequence of statements will cause the program to be executed and the results stored on a Fastrand file GPDALG*STDAT. Notice that program termination is governed by the stage (IOUT=1).

```
@ASG,A              GPDALG*STSPP.

@ASG,A              GPDALG*SPDAT.

@USE                10,GPDALG*STDAT.

@XQT                GPDALG*STSPP.MAP

$CONDAT

JIN=0, NCONTR=6, NX=6, MAXJ=10, VINIT=0., IOUT=1

ID='NTEST', XIN=10,0,0,0,0,0,0,

U(1,1)=2,3,3,3,3,3, U(1,2)=0,0,0,0,0,0,

U(1,3)=1,1,1,1,1,1, U(1,4)=0,2,2,2,2,2,

$END
```

The output of the program is a list of state numbers, costs, and the feasible states occurring in the final stage of the program, as illustrated in Table 4-4.

Now if the user desires to see the optimal path taken from the initial state to one or more of the final states (say state number 930 in Table 4-4). The following sequence of statements results in the desired backchaining:

```
@ASG,A              GPDALG*GPDFIL.

@ASG,A              GPDALG*SPDAT.

@USE                10,GPDALG*SPDAT

@XQT                GPDALG*GPDFIL.MAP

$CONDAT

  IOUT=5,

$END
```

TABLE 4-4

Feasible Final States for 10 Year Program

THE FOLLOWING ARE THE FINAL STATE VECTORS DETERMINED FOR PROGRAM NTEST WHICH MET FINAL CONDITIONS AT
TIME = 10

LINE	COST	X 1	X 2	X 3	X 4	X 5	X 6	X
900	11182.000	13	4	4	4	5	5	
901	11505.000	13	4	5	4	5	5	
902	10615.000	13	4	6	4	5	5	
903	11312.000	13	4	4	5	5	5	
904	11635.000	13	4	5	5	5	5	
905	10945.000	13	4	6	5	5	5	
906	11294.000	13	4	4	4	6	5	
907	11617.000	13	4	5	4	6	5	
908	10927.000	13	4	6	4	6	5	
909	11424.000	13	4	4	5	6	5	
910	11747.000	13	4	5	5	6	5	
911	11057.000	13	4	6	5	6	5	
912	11588.000	13	4	4	4	7	5	
913	11299.000	13	4	6	4	7	5	
914	11718.000	13	4	4	5	7	5	
915	11429.000	13	4	6	5	7	5	
916	11240.000	13	4	4	4	5	6	
917	11563.000	13	4	5	4	5	6	
918	10873.000	13	4	6	4	5	6	
919	11370.000	13	4	4	5	5	6	
920	11693.000	13	4	5	5	5	6	
921	11003.000	13	4	6	5	5	6	
922	11352.000	13	4	4	4	6	6	
923	11675.000	13	4	5	4	6	6	
924	10985.000	13	4	6	4	6	6	
925	11482.000	13	4	4	5	6	6	

Table 4-4 (Con't)

926	11805.000	13	4	5	5	6	6
927	11115.000	13	4	6	5	6	6
928	11357.000	13	4	6	4	7	6
929	11776.000	13	4	4	5	7	6
930	11487.000	13	4	6	5	7	6
931	11911.000	13	4	5	4	7	5
932	12041.000	13	4	5	5	7	5
933	12099.000	13	4	5	5	7	6
934	11646.000	13	4	4	4	7	6
935	11969.000	13	4	5	4	7	6

```
$PRTDAT

  NFIN=1, XFIN=930

$END
```

The output is again the feasible states presented in Table 4-4 and the particular optimal solution connecting the initial state with the specified final state. This solution is presented in Table 4-5.

The same problem is worked again with IOUT=3 which means that a constraint governs termination of the program. This first sequence of statements causes results to go on file 10.

```
@ASG,A          GPDALG*GPDFIL.

@ASG,A          GPDALG*SPDAT.

@USE            10,GPDALG*SPDAT.

@XQT            GPDALG*GPDFIL.MAP

$CONDAT

JIN=0, NCONTR=6, MAXJ=10, VINIT=0, IOUT=3,

ID='NTEST', XIN=10,0,0,0,0,0,

U(1,1)=2,3,3,3,3,3, U(1,2)=0,0,0,0,0,0,

U(1,3)=1,1,1,1,1,1, U(1,4)=2,2,2,2,2,2,

$END
```

The output is again a list of final feasible states that satisfy the terminal constraints.This output is given in Table 4-6.

Table 4-5

Optimal Solution for 10 Year Space Program

BEGIN BACKCHAIN AT TIME = 10 WITH SELECTED FINAL VECTOR

TIME	COST		X 1	X 2	X 3	X 4	X 5	X 6
10	11487.0	STATE	13	4	6	5	7	6
CONTROL			0	0	0	1	1	0
9	10594.0	STATE	13	4	6	4	6	6
CONTROL			0	0	0	1	1	1
8	9223.0	STATE	13	4	6	3	5	5
CONTROL			0	0	0	1	1	1
7	7927.0	STATE	13	4	6	2	4	4
CONTROL			0	0	1	1	1	1
6	6597.0	STATE	13	4	5	1	3	3
CONTROL			0	0	1	1	1	1
5	4956.0	STATE	13	4	4	0	2	2
CONTROL			0	1	1	0	1	1
4	3993.0	STATE	13	3	3	0	1	1
CONTROL			0	1	1	0	1	1
3	3152.0	STATE	13	2	2	0	0	0
CONTROL			1	1	1	0	0	0
2	2127.0	STATE	12	1	1	0	0	0
CONTROL			1	1	1	0	0	0
1	1016.0	STATE	11	0	0	0	0	0
CONTROL			1	0	0	0	0	0
0	.0	STATE	10	0	0	0	0	0

@ FIN

Table 4-6

Feasible Final States for Minimum Time Program

THE FOLLOWING ARE THE FINAL STATE VECTORS DETERMINED FOR PROGRAM NTEST WHICH MET FINAL CONDITIONS AT
TIME = 8

LINE	COST	X 1	X 2	X 3	X 4	X 5	X 6	X
1629	9890.000	13	4	6	0	7	5	
1630	9935.000	13	4	6	1	7	5	
1631	10070.000	13	4	6	2	7	5	
1632	10295.000	13	4	6	3	7	5	
1633	10595.000	13	4	6	4	7	5	
1634	10725.000	13	4	6	5	7	5	
1635	9948.000	13	4	6	0	7	6	
1636	9993.000	13	4	6	1	7	6	
1637	10128.000	13	4	6	2	7	6	
1638	10353.000	13	4	6	3	7	6	
1639	10653.000	13	4	6	4	7	6	
1640	10783.000	13	4	6	5	7	6	

@ FIN

Now the user wishes backchaining for a specific state, number 1640, in Table 4-6. The following sequence of statements will cause the backchaining:

```
@ASG,A          GPDALG*STSPP.
@ASG,A          GPDALG*STDAT.
@USE            10,GPDALG*STDAT.
@XQT            GPDALG*GPDFIL.MAP
$CONDAT
  IOUT=5
$END
$PRTDAT
  ID='NTEST'
$END
$PRTDAT
  XFIN=1640, NFIN=1,
$END
```

The output from execution again points out all the feasible final states given in Table 4-6 and the optimal solution from the initial to the specified final state presented in Table 4-7.

Table 4-7

<u>Optimal Solution for Minimum Time Space Program</u>

BEGIN BACKCHAIN AT TIME = 8 WITH SELECTED FINAL VECTOR

TIME	COST		X 1	X 2	X 3	X 4	X 5	X 6
8	10783.0	STATE	13	4	6	5	7	6
CONTROL			0	0	1	1	2	1
7	9083.0	STATE	13	4	5	4	5	5
CONTROL			0	0	1	1	1	1
6	7161.0	STATE	13	4	4	3	4	4
CONTROL			0	0	1	1	1	1
5	5438.0	STATE	13	4	3	2	3	3
CONTROL			0	1	1	1	1	1
4	4077.0	STATE	13	3	2	1	2	2
CONTROL			0	1	1	1	1	1
3	3140.0	STATE	13	2	1	0	1	1
CONTROL			1	1	1	0	1	1
2	2090.0	STATE	12	1	0	0	0	0
CONTROL			1	1	0	0	0	0
1	1016.0	STATE	11	0	0	0	0	0
CONTROL			1	0	0	0	0	0
0	.0	STATE	10	0	0	0	0	0

@ FIN

4.4 Discussion of Results

As was illustrated in the previous section, using the program is relatively easy once the problem is formulated in state space format. At the beginning of this section, two questions were posed: Whether a feasible solution exists, and if it does, what is the best solution?

To answer the first question, the time constraint of ten years was imposed and used as a terminating condition in the program. The program then generated all the optimal feasible states that are reachable at this stage. This data is presented in Table 4-4. Note that the desired final state (13, 4, 6, 5, 7, 6) is included in this set as state number 930. As in the "going home" example, more information than that requested is provided here. Again, this information might be very useful to the decision maker. If for example, at a later date it was discovered that the funding levels were lower than expected and the desired program cannot be completed, then an alternate, less ambitious program must be formulated. The investigator can then look at Table 4-4 and determine if any of the available final states meet his new cost constraint. If he finds one and is satisfied with the final status of the program, he can easily find the optimal schedule of that program by backchaining with that particular final state. The overall program need not be run. One can think of many ways to use this feature of dynamic programming.

The desired final state (13, 4, 6, 5, 7, 6) is used for backchaining. The output is given in Table 4-5. This optimal schedule is more clearly illustrated in Table 4-8. Note that the yearly expenditures satisfy both the lower and upper bounds. The lower bound of $1 billion was dropped after the first three years and the upper bound was increased to $2 billion after the fifth year. This illustrates the flexibility of the program and almost any type of constraint can be used.

The same problem was run again, except this time the desired final state was used as a stopping condition and time or the number of stages was left free. To our surprise, the desired final state was reached within only eight years and with smaller overall cost! The program outputted the feasible states at this stage as can be seen in Table 4-6. As in the previous case, this output can be useful to the investigator in evaluating his alternatives. The desired final state was then used for backchaining resulting in the output presented in Table 4-7. This optimal schedule is better illustrated in Table 4-9. Notice that the Shuttle development was accelerated in the last year. Again observe that all the constraints were satisfied. One can show that this is a minimum time schedule as well.

On comparing the two schedules, it is apparent that although the yearly expenditures for the minimum time schedule are higher, the fixed costs for the last two years are eliminated. This, then explains the corresponding lower cost.

This example was used only for illustrative purposes and to demonstrate the feasibility, flexibility and usefulness of the computer program. Many important factors were not considered. The costs of the various program segments, for example, are not independent and depend very much on the schedule itself. The costs used were in constant dollars while in fact dollar values change drastically in a time period of ten years. Some of these and other factors can be included in the program to give a more realistic simulation.

It is conceivable that for a class of problems, such as space program planning, an interface computer program can be built between the user and the basic dynamic programming package. In this interface program all the inputs to the basic program will be fixed except for a relatively small number of physically meaningful parameters which are left to be specified by the user. Such a program can then be used on an online terminal.

Table 4-8

Optimal Development Schedule

(For 10 Year Program)

Current Year	71	72	73	74	75	76	77	78	79	80
Apollo	Δ	Δ	Δ							
Skylab I		Δ	Δ	Δ	Δ					
Skylab II		Δ	Δ	Δ	Δ	Δ	Δ			
Skylab III						Δ	Δ	Δ	Δ	Δ
Shuttle				Δ	Δ	Δ	Δ	Δ	Δ	Δ
ILV				Δ	Δ	Δ	Δ	Δ	Δ	
Yearly Expenditures (In Billions)	1.016	1.111	1.025	0.841	0.963	1.641	1.330	1.296	1.371	0.893
Total Cost	1.016	2.127	3.152	3.993	4.956	6.597	7.927	9.223	10.594	11.487

Δ = 1 Year of Normal Development

Table 4-9

Optimal Development Schedule

(No Time Constraint)

Current Year	71	72	73	74	75	76	77	78	79	80
Apollo	Δ	Δ	Δ							
Skylab I		Δ	Δ	Δ	Δ					
Skylab II			Δ	Δ	Δ	Δ	Δ	Δ		
Skylab III				Δ	Δ	Δ	Δ	Δ		
Shuttle			Δ	Δ	Δ	Δ	Δ	ΔΔ		
ILV			Δ	Δ	Δ	Δ	Δ	Δ		
Yearly Expenditure	1.016	1.074	1.050	0.937	1.361	1.623	1.922	1.700		
Total Cost	1.016	2.090	3.140	4.077	5.438	7.161	9.083	10.783		

Δ = 1 Year of Normal Development

5.0 Summary

The Dynamic Programming Concept for multi-stage decision processes was explained via a very simple example. The type of problems for which this approach is useful were put in a general format known as the State Space Format. The problem of dimensionality associated with Dynamic Programming was substantially reduced by designing a procedure whereby only feasible states are considered. A computer program was developed using this procedure. In theory, it can solve any problem that can be put in the State Space Format, in practice, however, the program is limited as to the size of the problem it can handle. At any instant of time while generating the optimal solutions, the feasible states of at least two successive stages must be available in the computer core. This is the most important limitation of the program and at present the number of feasible states for any two successive stages is limited to 2500. The number of state variables should also be made as small as possible otherwise excessive computing time may result.

A realistic space program planning application was then formulated and put in state space format. A fixed time as well as minimum time program planning schedule problems were solved. The computer program was used to generate classes of optimal solutions as well as two particular solutions. The feasibility and usefulness of the concept as well as the computer program were demonstrated.

John E. Nahra

J. E. Nahra

Marjorie Odle

M. P. Odle

1015-MPO
1032-JEN-cp

Attachments
References
Appendix

REFERENCES

1. Dreyfus, Stuart E., Dynamic Programming and the Calculus of Variations, Academic Press, New York 1965.

2. Nahra, J. E., "Programming of the Optimal Evaluator Subprogram", Addressed Memorandum to C. L. Davis, May 14, 1970.

3. FORTRAN V, Programmer's Reference Manual, UNIVAC

GPDFIL.GPALG,GPALG
D COMPILED BY 1201 BC57E ON 20 JAN 71 AT 14:06:09.

PROGRAM

GE USED: CODE(1) 001421; DATA(0) 144511; BLANK COMMON(2) 000000

ON BLOCKS:

LOKPER 000064
PARAM 000025

NAL REFERENCES (BLOCK, NAME)

PACK
ENTER
COMPER
CONSTR
COST
COSLIM
NEXTST
COMCOS
SEARCH
POFILE
UNPCKS
PRTOUT
FINVEC
PRTOUF
PRTCHN
NINTR$
NRNL$
NERR2$
NWDU$
NIO1$
NIO2$
NWBU$
NREW$
NRBU$
NWEF$
NSTOP$

GE ASSIGNMENT (BLOCK, TYPE, RELATIVE LOCATION, NAME)

000073 12L	0001	000121 14L	0001	000033 142G	0001	000145 1
000311 25L	0001	000503 262G	0001	000506 264G	0001	000540 3
000645 333G	0001	000657 343G	0001	000664 347G	0001	000344 3
000356 36L	0001	000715 363G	0001	000721 367G	0001	000373 3
001024 421G	0001	001060 436G	0001	001074 446G	0001	001101 4
001137 467G	0001	001143 473G	0001	000434 50L	0001	000405 5
001167 506G	0001	000441 52L	0001	001263 544G	0001	000523 5
001335 572G	0001	001341 576G	0001	000612 58L	0001	000702 5
002710 6002F	0000	002716 6003F	0000	002743 6004F	0001	001360 6
001030 63L	0001	001117 65L	0001	001204 70L	0001	001211 7
002730 76F	0001	001301 80L	0001	001320 811L	0001	001405 8

```
R 000000 COST      0000 I 000360 CP       0000 I 000334 CS       0000 L 002556 FI
I 002571 ID        0000 I 002630 ID1      0000 I 002616 IFJ      0000 I 002626 II
I 002615 IJ        0000 I 002625 IJM1     0003 I 000002 IOKPER   0000 I 002570 IO
I 002520 IT        0000 I 002624 J        0000 I 002563 JIN      0000 I 002614 JJ
I 002603 JJJJJ     0000 I 002573 JT       0004 I 000000 JTIME    0000 I 002631 JT
L 002560 KF        0000 I 002606 KIND     0000 I 002575 KOUNT    0000 L 002561 KP
I 002601 LPST      0000 I 002620 LPST2    0000 I 002566 MAXJ     0000 I 000454 MA
I 002612 MJ2       0000 I 002564 NCONTR   0000 I 002572 NFIN     0000 I 002627 NI
I 002617 NJ        0003 I 000001 NONPER   0000 I 002574 NPAGE    0000 I 002602 NP
I 000000 NTPER     0000 I 002622 NTR      0000 I 002565 NX       0000 R 002604 P
I 000404 PST       0000 R 002632 SKIP     0000 I 000430 ST       0000 L 002557 TA
R 002600 V         0000 R 002567 VINIT    0000 R 002605 VV       0000 I 002753 X
I 000001 XIN       0000 R 007657 XV
```

```
      DIMENSION U(20,11),CS(20),CP(20),PST(20),ST(20),XIN(20),
     .MAXXJ(20,2),IOKPER(50),XFIN(1000),XV(2000),IP(20)
      DIMENSION IT(20)
      DIMENSION LENGP(10)
      INCLUDE DECL,LIST
        PARAMETER MAXX=2500
        DIMENSION X(MAXX,20)
  END
      EQUIVALENCE(XV(1),X(1,2))
      COMMON/LOKPER/NTPER,NONPER,IOKPER
      COMMON/PARAM/JTIME,XIN
      INTEGER U,X,CS,CP,PST,ST,XIN,XFIN
      LOGICAL FIRST,TABFLG,KF,KPERM
      LOGICAL PRFLG
      DATA PRFLG/.TRUE./
      NAMELIST/CONDAT/JIN,XIN,U,NCONTR,NX,MAXJ,VINIT,IOUT,
     .KPERM,ID
C
C        INPUT VIA CONDAT
C             JIN                      INITIAL TIME J
C             XIN                      VECTOR OF INITIAL STATE CONFIG
C             U                        MATRIX OF VALUES OF CONTROL VBLS
C             NCONTR                   NUMBER OF CONTROLS
C             NX                       NUMBER OF COMPONENTS OF STATE VECTOR
C             MAXJ                     MAXIMUM TIME ALLOWABLE
C             VINIT                    INITIAL COST V
C             IOUT                     OUTPUT INDICATOR
C             =1  PROBLEM TERMINATION CONTROLLED BY JTIME REACHING
C                 MAXJ.  PRINT STATES AT TIME MAXJ AND EXIT.  STORE
C                 ENTIRE RESULTS ON FILE 10.
C             =2  SAME AS 1 EXCEPT REQUEST STATE NUMBER(S) FOR BACK-
C                 CHAIN BEFORE EXIT.
C             =3  PROBLEM TERMINATION CONTROLLED BY FINAL CONSTRAINTS.
C                 PRINT STATES WHEN CONDITION SATISFIED AND EXIT.
C                 STORE ENTIRE RESULTS ON FILE 10.
C             =4  SAME AS 3 EXCEPT REQUEST STATE NUMBER(S) FOR BACK-
C                 CHAIN BEFORE EXIT.
C             =5  PROBLEM RESULTS ALREADY ON FILE 10.  REQUEST STATES
C                 FOR BACKCHAINING.
C             KPERM                    FLAG=TRUE, KEEP FEASIBLE PERMUTATIONS
```

```
C                                                 GENERATED FOR TABLE LOOKUP
C                                               =FALSE, DONOT KEEP PERMUTATIONS
C              ID                       PROBLEM IDENTIFIER, 6 ALPHABETIC CHAR
C
               NAMELIST/PRTDAT/ID,XFIN,NFIN
C       INPUT VIA PRTDAT
C              ID                       PROBLEM IDENTIFIER FOR PRINTING
C              XFIN                     VECTOR OF PLACE IN STATE ARRAY OF
C                                       SOLUTIONS ARE INTERESTED IN SEEING
C              NFIN                     NUMBER OF FINAL VECTOR TRACES, I E
C                                       NUMBER OF INDICES IN XFIN
C
C
C       INITIALIZATION
C
  1            JT=1
               NPAGE=1
               KOUNT=0
               NTPER=1
               MAXXJ(JT,1)=1
               MAXXJ(JT,2)=1
               FIRST=.TRUE.
               NIPTRJ=1
C
C       INPUT SECTION
C
               READ(5,CONDAT,END=6050)
               IF (IOUT-5)5,70,70
  5            JTIME=JIN
               DO 10  I=1,NX
  10           CS(I)=XIN(I)
               V=VINIT
               CALL PACK(CS,CP,PST,NX,NCONTR,LPST)
               KOUNT=KOUNT+1
               CALL ENTER(X,PST,VINIT,1,0,LPST,NPAGE)
                    XV(1)=VINIT
               MAXXJ(JT+1,1)=MAXXJ(JT,2)+1
               MAXXJ(JT+1,2)=MAXXJ(JT,2)
C
C       COMPUTE A PERMUTATION OF U
C
  12           CALL COMPER(IP,U,NCONTR,TABFLG,FIRST,NPER,$5000)
               DO 13  I=1,NCONTR
               JJJJJ=IP(I)
  13           CP(I)=U(I,JJJJJ)
C
C       AT JTIME=JIN TEST FEASIBILITY OF INITIAL STATE AND PERMUTATION
C
  14           IF (JTIME .GT. JIN)  GO TO 15
               CALL CONSTR(CS,CP,NX,NCONTR,$35)
C
C       INITIAL STATE AND PERMUTATION SATISFY CONSTRAINTS.  IF KPERM
C       IS TRUE, SAVE THIS INDEX NPER AS THE NUMBER OF A FEASIBLE PER-
C       MUTATION
C
               IF ( .NOT. KPERM)  GO TO 15
               IOKPER(NTPER)=NPER
```

```
                NTPER=NTPER+1
      C
      C     AT 15, ALL CONSTRAINTS OF THIS STATE AND PERMUTATION ARE SAT-
      C     ISFIED.  COMPUTE COST P.
      C
       15       P=COST(CS,CP,NX,NCONTR)
      C
      C     TEST FEASIBILITY OF P
      C
                CALL COSLIM(P,$35)
      C
      C     COST CONSTRAINT SATISFIED, DERIVE NEXT STATE
      C
                CALL NEXTST(CS,CP,ST,NX,NCONTR)
      C
      C     NEW STATE IN VECTOR ST.  TEST FEASIBILITY
      C
                CALL CONSTR(ST,CP,NX,NCONTR,$35)
      C
      C     NEW STATE SATISFACTORY, COMPUTE V ASSOCIATED WITH ST AND CP.
      C     ALSO TEST FEASIBILITY
      C
                CALL COMCOS(CS,CP,ST,P,V,VV,$35)
      C
      C     COST V ACCEPTABLE.  SEARCH LIST OF STATES AND CONTROL CONFIGU
      C     RATIONS AT THAT JTIME TO SEE IF THIS STATE ALREADY CONSIDERED.
      C
                CALL SEARCH(X,MAXXJ(JT+1,1),MAXXJ(JT+1,2),ST,NX,KF,KIND)
      C
      C     KF TRUE MEANS ST EXISTS ON LIST X
      C
                IF (KF)  GO TO 25
      C
      C     ST CONFIGURATION DOES NOT APPEAR ON LIST X.  ENTER IT WITH THE
      C     CORRESPONDING PERMUTATION CP AND V AND LINK TO PREVIOUS STATE
      C
                CALL PACK(ST,CP,PST,NX,NCONTR,LPST)
                MAXXJ(JT+1,2)=MAXXJ(JT+1,2)+1
                KOUNT=KOUNT+1
                IF (KOUNT .GT. MAXX) CALL POFILE(JT,NPAGE,LENGP,X,MAXXJ,
           .    LPST,KOUNT,NIPTRJ,$6050)
                CALL ENTER(X,PST,VV,MAXXJ(JT+1,2),NIPTRJ,LPST,NPAGE)
                     JJJ=MAXXJ(JT+1,2)
                     XV(JJJ)=VV
                GO TO 35
      C
      C     STATE ST APPEARS ON LIST X.  CHECK FOR BEST COST V.  IF NEW ONE
      C     IS BETTER, CHANGE PACKED U TO CP AND STORE NEW COST AND LINK
      C
       25       BV=XV(KIND)
                IF (VV .GE. BV)  GO TO 35
                CALL PACK(ST,CP,PST,NX,NCONTR,LPST)
                CALL ENTER(X,PST,VV,KIND,NIPTRJ,LPST,NPAGE)
                     XV(KIND)=VV
      C
      C     GO TO NEXT STATE AT TIME JTIME
      C
```

```
      35          NIPTRJ=NIPTRJ+1
                  IF (NIPTRJ .LE. MAXXJ(JT,2))   GO TO 37
C
C       EXHAUSTED ALL STATES AT LEVEL JTIME FOR THIS CP.  GET NEXT
C       PERMUTATION AFTER REINITIALIZING THE JTIME BLOCK OF X.S
C
      36          NIPTRJ=MAXXJ(JT,1)
                  CALL UNPCKS(X,NIPTRJ,CS,NX)
                  V=XV(NIPTRJ)
                  GO TO 12
C
C       PUT NEW STATE FROM X INTO CS
C
      37          CALL UNPCKS(X,NIPTRJ,CS,NX)
                  V=XV(NIPTRJ)
                  GO TO 14
C
C       AT 5000 ALL PERMUTATIONS EXHAUSTED FOR TIME JTIME.  ALL NEW X'S
C       GENERATED AND TESTED AND STORED FOR TIME JTIME+1. BEGIN WORK
C       ON NEW JTIME
C
      5000        IF (JTIME .GT. JIN)  GO TO 40
C
C       AT INITIAL TIME - MIGHT WANT TO SAVE FEASIBLE PERMUTATIONS IN
C       TABLE
C
                  IF (KPERM)  TABFLG=.TRUE.
      40          JTIME=JTIME+1
                  GO TO (50,50,60,60,70),IOUT
C
C       IOUT=1 OR IOUT=2. MAXJ SPECIFIED - TEST IF MAX STAGE EXCEEDED
C
      50          IF (JTIME .GE. MAXJ)  GO TO 55
C
C       NOT YET AT MAX TIME - GO ON TO NEXT TIME BLOCK
C
      52          JT=JT+1
                  MAXXJ(JT+1,1)=MAXXJ(JT,2)+1
                  MAXXJ(JT+1,2)=MAXXJ(JT,2)
                  FIRST=.TRUE.
                  NONPER=1
                  MJ1=MAXXJ(JT,1)
                  MJ2=MAXXJ(JT,2)
                  IF (PRFLG)
     .            WRITE(6,6002)JT,JTIME,MJ1,MJ2,((X(III,JJ),JJ=1,7),
     .            III=MJ1,MJ2)
      6002        FORMAT(1H0,'MAIN',4I8/(1X,7(O12,2X)))
                  GO TO 36
C
C       JTIME GTR THAN MAXJ
C       OUTPUT ALL FINAL VECTORS AT JTIME LEVEL
C
      55          IJ=MAXXJ(JT+1,1)
                  IFJ=MAXXJ(JT+1,2)
                  LENGP(NPAGE)=IFJ
                  NJ=IFJ-IJ+1
                  DO 56  I=1,NJ
```

```
      56       XFIN(I)=IJ+I-1
               CALL PRTOUT(X,JTIME,XFIN,NJ,NX,ID,MAXXJ,JT+1)
               WRITE(10) ID
               LPST2=LPST+2
               NPM1=NPAGE-1
               IF (NPM1 .EQ. 0)  GO TO 58
               NTR=0
               DO 57 I=1,NPM1
      57       NTR=NTR+LENGP(I)
      58       WRITE(10)JTIME,LPST2,NX,NCONTR,NPAGE,NTR,IJ,(LENGP(K),
     .         K=1,NPAGE)
               IF (NPM1 .EQ. 0)  GO TO 59
               REWIND 3
               DO 581 I=1,NTR
               READ(3) (IT(K),K=1,LPST2)
      581      WRITE(10) (IT(K),K=1,LPST2)
      59       DO 591 I=1,IFJ
      591      WRITE(10) (X(I,J),J=1,LPST2)
      C
      C        READY FOR OUTPUT SEQUENCES
      C
               END FILE 10
               IF (IOUT .EQ. 1)  GO TO 6050
               GO TO 70
      C
      C        IOUT=3 OR IOUT=4. FINAL CONDITIONS SPECIFIED - PRINT OUT ALL STATE
      C        VECTORS THAT SATISFY THE FINAL CONDITIONS
      C
      60       CALL FINVEC(X,JT,MAXXJ,NX,XFIN,NJ)
               IF (NJ .EQ. 0)  GO TO 52
               CALL PRTOUT(X,JTIME,XFIN,NJ,NX,ID,MAXXJ,JT+1)
               WRITE(10) ID
               LPST2=LPST+2
               MAXXJ(JT+1,2)=NJ+MAXXJ(JT+1,1)-1
               IFJ=MAXXJ(JT+1,2)
               IJ=MAXXJ(JT+1,1)
               LENGP(NPAGE)=IFJ
               NPM1=NPAGE-1
               IF (NPM1 .EQ. 0)  GO TO 63
               NTR=0
               DO 62  I=1,NPM1
      62       NTR=NTR+LENGP(I)
      63       WRITE(10)JTIME,LPST2,NX,NCONTR,NPAGE,NTR,IJ,(LENGP(K),
     .         K=1,NPAGE)
               IF (NTR .EQ. 0)  GO TO 65
               REWIND 3
               DO 64 I=1,NTR
               READ(3) (IT(K),K=1,LPST2)
      64       WRITE(10) (IT(K),K=1,LPST2)
      65       IJM1=IJ-1
               DO 66  I=1,IJM1
      66       WRITE(10) (X(I,J),J=1,LPST2)
               DO 67  I=1,NJ
               II=XFIN(I)
      67       WRITE(10) (X(II,J),J=1,LPST2)
      C
      C        READY FOR OUTPUT SEQUENCE TRACE
```

```
C
                 END FILE 10
                 IF (IOUT .EQ. 3) GO TO 6050
C
C       IOUT=3,4,ORS.  ACCEPT INPUT OF INDICES OF STATES AS FINAL COND-
C       ITIONS THAT ARE TO BE DISPLAYED
C
C       READ PRTDAT FOR THE OUTPUT INFORMATION
C
 70              WRITE(6,6003)
 6003            FORMAT(1H1,'WHICH PROBLEM ID IS OF INTEREST FOR BACKCHAIN
     .ING')
 71              READ(5,PRTDAT,END=6050)
                 IF (NID .EQ. ID) GO TO 85
                 NID=ID
                 REWIND 10
C
C       ID IN PRTDAT SPECIFIES WHICH PROBLEM INTERESTED IN.  ALL ARE ON
C       FILE 10 THEREFORE MUST SEARCH FOR RIGHT ID
C
 75              READ(10,END=751) ID1
                 READ(10) JT1,LPST2,NX,NCONTR,NPAGE,NTR,IJ,(LENGP(K),
     .           K=1,NPAGE)
                 IF (ID1 .EQ. ID) GO TO 80
 751             WRITE(6,76)
 76              FORMAT(1H0,'  WRONG PROBLEM ID,  THAT PROBLEM NOT STORED
     .ON FILE 10')
                 GO TO 6050
 80              LPST=LPST2-2
                 IF (NTR .EQ. 0)  GO TO 811
                 DO 81  I=1,NTR
 81              READ(10) SKIP
 811             IFJ=LENGP(NPAGE)
                 DO 82  I=1,IFJ
 82              READ(10) (X(I,J),J=1,LPST2)
                 NJ=IFJ-IJ+1
                 DO 83  I=1,NJ
 83              XFIN(I)=IJ+I-1
                 CALL PRTOUF(X,JT1,XFIN,NJ,NX,ID1)
                 WRITE(6,6004)
 6004            FORMAT(1H0,'   NOW INPUT INDICES FOR BACKCHAINING')
                 GO TO 71
 85              CALL PRTCHN(X,JT1,XFIN,NFIN,NX,NCONTR,NPAGE,LENGP)
 6050            CONTINUE
                 END

D OF COMPILATION:        NO  DIAGNOSTICS.
```

```
      SEARCH,SEARCH
)0 COMPILED BY 1201 BCS7E ON 07 JAN 71 AT 14:09:10.

)UTINE SEARCH     ENTRY POINT 000077

AGE USED: CODE(1) 000125; DATA(0) 000054; BLANK COMMON(2) 000000

RNAL REFERENCES (BLOCK, NAME)

3    PACK
4    NERR3$

AGE ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

1    000051 10L        0001    000030 116G       0001    000040 121G       0001    000057 3(
0    000034 INJP$      0000 I 000031 J           0000 I 000027 LT          0000 I 000024 N\
0 I 000000 T           0000 I 000026 T1
```

```
                 SUBROUTINE SEARCH(X,MJ1,MJ2,ST,NX,KF,KIND)
                 IMPLICIT INTEGER (A-Z)
                 INCLUDE DECL,LIST
                   PARAMETER MAXX=2500
                   DIMENSION X(MAXX,20)
       END
                 DIMENSION ST(1),T(20)
      C
      C    SEARCH ARRAY X FOR STATE ST ONLY AT STATES GENERATED DURING
      C    TIME JTIME AS DETERMINED BY MAXXJ(JT+1,1)...MAXXJ(JT+1,2)
      C
                 LOGICAL KF
                 KF=.FALSE.
                 NWFS=(NX-1)/4+1
                 NWFSP2=NWFS+2
                 CALL PACK(ST,T1,T,NX,1,LT)
                 DO 10  I=MJ1,MJ2
                 DO 20  J=1,NWFS
                 IF (T(J) .NE. X(I,J+2)) GO TO 10
       20        CONTINUE
      C
      C    THERE IS A MATCH IN STATES AT STATE I.  GO TO 30
      C
                 GO TO 30
       10        CONTINUE
      C
      C    NO MATCH.  RETURN 0
      C
                 KIND=0
                 RETURN
       30        KF=.TRUE.
                 KIND=I
```

```
      RETURN
      END
END OF COMPILATION:      NO  DIAGNOSTICS.
```

```
      COMPER,COMPER
)0 COMPILED BY 1201 BCS7E ON 07 JAN 71 AT 14:09:03.

)UTINE COMPER    ENTRY POINT 000216

AGE USED: CODE(1) 000244; DATA(0) 000026; BLANK COMMON(2) 000000

MON BLOCKS:

3    LOKPER 000003

RNAL REFERENCES (BLOCK, NAME)

4    NERR4$
5    NERR3$

AGE ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

1     000027 10L       0001   000171 100L     0001   000173 110L     0001   000020 1
1     000112 146G      0001   000137 161G     0001   000145 164G     0001   000062 2
1     000120 60L       0001   000123 65L      0001   000165 85L      0000 I 000000 I
3 I   000002 IOKPER    0000 I 000001 IQ       0000 I 000005 JJ       0000 I 000003 N
0 I   000002 NT        0003 I 000000 NTPER    0000 I 000004 NUP
```

```
C
C       FOR COMPUTING PERMUTATIONS OR FOR ACCESSING NEXT FEASIBLE PERM
C
              SUBROUTINE COMPER(NCP,U,NC,TABFLG,FIRST,NPER,$)
              IMPLICIT INTEGER (A-Z)
              LOGICAL FIRST,TABFLG
              DIMENSION NCP(1),U(20,11),IOKPER(1)
              COMMON/LOKPER/NTPER,NONPER,IOKPER
              IF (TABFLG)  GO TO 50
              NONPER=1
C
C       NO PABLE LOOKUP OR FIRST TIME GENERATING PERMUTATIONS
C
              IF ( .NOT. FIRST)  GO TO 10
              NPER=1
              DO 5  I=1,NC
  5           NCP(I)=2
              FIRST=.FALSE.
              RETURN
  10          NPER=NPER+1
              DO 20  IQ=1,NC
              NT=NCP(IQ)+1
              IF (NT .LE. U(IQ,1)+1)  GO TO 25
              NCP(IQ)=2
  20          CONTINUE
C
```

```
C        CONSIDERED ALL PERMUTATIONS
C
                    RETURN 7
 25                 NCP(IQ)=NT
                    RETURN
 50                 IF (NONPER .GT. NTPER) RETURN 7
                    IF (NONPER .GT. 1)  GO TO 60
                     DO 51 I=1,NC
 51                 NCP(I)=2
                    NLOW=1
                    GO TO 65
 60                 NLOW=IOKPER(NONPER-1)
 65                 NUP=IOKPER(NONPER)-1
                    IF (NUP-NLOW .LT. 0)  GO TO 110
                    DO 100  JJ=NLOW,NUP
                    DO 80  IQ=1,NC
                    NT=NCP(IQ)+1
                    IF ( NT .LE. U(IQ,1)+1)  GO TO 85
                    NCP(IQ)=2
 80                 CONTINUE
                    GO TO 100
 85                 NCP(IG)=NT
 100                CONTINUE
 110                NONPER=NONPER+1
                    RETURN
                    END
```

D OF COMPILATION: NO DIAGNOSTICS.

```
     ENTER,ENTER
 COMPILED BY 1201 BCS7E ON 07 JAN 71 AT 14:08:11.

TINE ENTER     ENTRY POINT 000044

E USED: CODE(1) 000060; DATA(0) 000023; BLANK COMMON(2) 000000

AL REFERENCES (BLOCK, NAME)

  NERR3$

E ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

  000023 113G      0000 I 000000 I         0000   000005 INJP$     0000 I 000001 LP
```

```
C
C       SUBROUTINE ENTER PUTS A ROW IN ARRAY X AT THE PLACE SPECIFIED
C       BY LINO.  THE INFO ENTERED IS IN PST OF LENGTH LPST. FIRST WORD OF
C       X I.E. X(LINO,1) IS BACK LINK.  2ND WORD, X(LINO,2) IS V.
C       X(LINO,3) ... X(LINO,3+LPST) IS PST.
C
C
              SUBROUTINE ENTER(X,PST,V,LINO,LINK,LPST,NPAGE)
              IMPLICIT INTEGER (A-U,W-Z)
              INCLUDE DECL,LIST
                PARAMETER MAXX=2500
                DIMENSION X(MAXX,20)
  END
              DIMENSION PST(1)
              FLD(0,18,X(LINO,1))=NPAGE
              FLD(18,18,X(LINO,1))=LINK
              DO 10  I=1,LPST
   10         X(LINO,I+2 ) =PST(I)
              LPST2=LPST+2
              RETURN
              END
```

```
ID OF COMPILATION:        NO  DIAGNOSTICS.
```

```
      PACK,PACK
' COMPILED BY 1201 BCS7E ON 07 JAN 71 AT 14:07:30.

ITINE PACK      ENTRY POINT 000205

,E USED: CODE(1) 000232; DATA(0) 000047; BLANK COMMON(2) 000000

IAL REFERENCES (BLOCK, NAME)

  NERR3$

,E ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

  000055 110G       0001    000121 122G       0000 I 000002 I          0000    000007 IN
I 000001 NWFP       0000 I 000000 NWFS       0000 I 000003 NWFSP1


      C
      C       SUBROUTINE PACK PACKS CURRENT STATE VECTOR AND CURRENT PERMUTATION
      C       VECTOR INTO 1/4 OF SIZE - 4 INDICES PER WORD..  THE FIRST
      C       (NX-1)/4+1 WORDS ARE THE STATE VECTOR, THE NEXT NC-1/4+1 WORDS
      C       ARE THE CONTROL VECTOR.
      C
                    SUBROUTINE PACK(CS,CP,PST,NX,NC,LPST)
                    IMPLICIT INTEGER (A-Z)
                    DIMENSION CS(NX),CP(NC),PST(1)
                    NWFS=(NX-1)/4+1
                    NWFP=(NC-1)/4+1
                    DO 10  I=1,NWFS
                    FLD(0,9,PST(I))=CS(4*I-3)
                    FLD(9,9,PST(I))=CS(4*I-2)
                    FLD(18,9,PST(I))=CS(4*I-1)
                    FLD(27,9,PST(I))=CS(4*I)
         10         CONTINUE
                    NWFSP1=NWFS+1
                    DO 20  I=1,NWFP
                    J=I+NWFSP1  -1
                    FLD(0,9,PST(J))=CP(4*I-3)
                    FLD(9,9,PST(J))=CP(4*I-2)
                    FLD(18,9,PST(J))=CP(4*I-1)
                    FLD(27,9,PST(J))=CP(4*I)
         20         CONTINUE
                    LPST=NWFS+NWFP
                    RETURN
                    END

ID OF COMPILATION:        NO  DIAGNOSTICS.
```

```
     UNPCKS,UNPCKS
  COMPILED BY 1201 BCS7E ON 07 JAN 71 AT 14:07:21.

 TINE UNPCKS    ENTRY POINT 000074

 E USED: CODE(1) 000111; DATA(0) 000030; BLANK COMMON(2) 000000

 IAL REFERENCES (BLOCK, NAME)

   NERR3$

 E ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

   000036 112G      0000 I 000001 I         0000   000005 INJP$     0000 I 000000 NWI


 C
 C       SUBROUTINE UNPCK EXTRACTS THE STATE VECTOR FORM ARRAY X, THE
 C       NIP TH ROW.
 C
                 SUBROUTINE UNPCKS(X,NIP,CS,NX)
                 INCLUDE DECL,LIST
                   PARAMETER MAXX=2500
                   DIMENSION X(MAXX,20)
  END
                 DIMENSION CS(1)
                 IMPLICIT INTEGER(A-Z)
                 NWFS=(NX-1)/4+1
                 DO 10  I=1,NWFS
                 CS(4*I-3)=FLD( 0,9,X(NIP,I+2))
                 CS(4*I-2)=FLD( 9,9,X(NIP,I+2))
                 CS(4*I-1)=FLD(18,9,X(NIP,I+2))
                 CS(4*I)  =FLD(27,9,X(NIP,I+2))
  10             CONTINUE
                 RETURN
                 END

 ID OF COMPILATION:        NO  DIAGNOSTICS.
```

```
     UNPCKP,UNPCKP
 COMPILED BY 1201 BCS7E ON 07 JAN 71 AT 14:03:55.

TINE UNPCKP    ENTRY POINT 000100

E USED: CODE(1) 000114; DATA(0) 000026; BLANK COMMON(2) 000000

AL REFERENCES (BLOCK, NAME)

  NERR3$

E ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

  000043 113G      0000 I 000002 I         0000    000006 INJP$     0000 I 000001 NWI


  1*                     SUBROUTINE UNPCKP(X,NIP,CP,NC,NX)
  2*                     INCLUDE DECL,LIST
  2*                       PARAMETER MAXX=2500
  2*                       DIMENSION X(MAXX,20)
  2*      END
  3*                     DIMENSION CP(1)
  4*                     IMPLICIT INTEGER (A-Z)
  5*                     NWFS=(NX-1)/4+1
  6*                     NWFP=(NC-1)/4+1
  7*                     DO 10  I=1,NWFP
  8*                     CP(4*I-3)=FLD( 0,9,X(NIP,I+NWFS+2))
  9*                     CP(4*I-2)=FLD( 9,9,X(NIP,I+NWFS+2))
 10*                     CP(4*I-1)=FLD(18,9,X(NIP,I+NWFS+2))
 11*                     CP(4*I)=  FLD(27,9,X(NIP,I+NWFS+2))
 12*      10             CONTINUE
 13*                     RETURN
 14*                     END
ID OF COMPILATION:          NO  DIAGNOSTICS.
```

```
     POFILE,POFILE
 COMPILED BY 1201 BCS7E ON 07 JAN 71 AT 14:08:55.


TINE POFILE    ENTRY POINT 000232


E USED: CODE(1) 000273; DATA(0) 000077; BLANK COMMON(2) 000000


AL REFERENCES (BLOCK, NAME)

  NWBU$
  NIO1$
  NIO2$
  NWDU$
  NERR4$
  NERR3$


E ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

  000007 100F      0001    000200 1000L     0000    000014 1001F     0001    000052 11
  000115 137G      0001    000142 147G      0001    000143 152G      0000 I 000002 I
  000045 INJP$     0000 I 000005 IS        0000 I 000006 ISS       0000 I 000003 J
I 000000 LPST2
```

```
                    SUBROUTINE POFILE(JT,NPAGE,LENGP,X,MAXXJ,LPST,KOUNT,
                    NIP,$)
                    INCLUDE DECL,LIST
                      PARAMETER MAXX=2500
                      DIMENSION X(MAXX,20)
       END
                    DIMENSION MAXXJ(20,2),LENGP(1)
      C
      C       STARTING WITH INDEX 1 PUT OUT UP TO MAXXJ(JT-1,2)
      C
                    IF (JT-1 .EQ. 0)  GO TO 1000
                    LPST2=LPST+2
                    LENGP(NPAGE)=MAXXJ(JT-1,2)
                    LP=LENGP(NPAGE)
                    DO 10  I=1,LP
       10           WRITE(3) (X(I,J),J=1,LPST2)
                    WRITE(6,100) NPAGE,LENGP(NPAGE)
       100          FORMAT(1H0,'  IN POFILE ',2I8)
      C
      C       FIX LINKS AND PAGE NUMBER IN REMAINDER
      C
                    IFJ=MAXX-MAXXJ(JT,1)+1
                    IS=MAXXJ(JT+1,1)
                    NPAGE=NPAGE+1
                    DO 15  I=IS,MAXX
                    FLD(0,18,X(I,1))=NPAGE
                    FLD(18,18,X(I,1))=FLD(18,18,X(I,1))-LP
```

```
 15           CONTINUE
C
C      MOVE STATES GENERATED AT TIME JT AND JT+1
C
              ISS=MAXXJ(JT,1)
              DO 20  I=ISS,MAXX
              DO 20  J=1,LPST2
 20           X(I-ISS+1,J)=X(I,J)
C
C      FIX MAXXJ ARRAY TO REFLECT THIS CHANGE
C
              MAXXJ(1,1)=1
              MAXXJ(1,2)=MAXXJ(JT,2)-LP
              MAXXJ(2,1)=MAXXJ(1,2)+1
              MAXXJ(2,2)=MAXX-LP
              KOUNT=MAXXJ(2,2)
              NIP=NIP-LP
              JT=1
              RETURN
 1000         WRITE(6,1001)
 1001         FORMAT(1H0,'  THIS PROBLEM CANNOT BE CONTINUED BECAUSE MO
     .RE THAN 2000 STATES WERE GENERATED DURING ONE TIME SLOT.')
              RETURN 9
              END
```

OF COMPILATION: NO DIAGNOSTICS.

```
    GPDFIL.PRTOUT,PRTOUT
OMPILED BY 1201 BCS7E ON 18 JAN 71 AT 10:46:18.

NE PRTOUT     ENTRY POINT 000233
   PRTOUF     ENTRY POINT 000265

USED: CODE(1) 000313; DATA(0) 000130; BLANK COMMON(2) 000000

 REFERENCES (BLOCK, NAME)

UNPCKS
NWDU$
NIO1$
NIO2$
NERR3$

ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

000032 100F        0000    000072 101F        0000    000065 102F        0001    000026 120G
000074 140G        0001    000132 160G        0001    000142 165G        0001    000173 176G
000027 II          0000    000100 INJP$       0000 I  000031 J           0000 I  000026 K
000000 NT          0000 R  000030 XV


*                      SUBROUTINE PRTOUT(X,JT,XF,NJ,NX,ID,MAXXJ,JTTT)
*                      INCLUDE DECL,LIST
*                        PARAMETER MAXX=2500
*                        DIMENSION X(MAXX,20)
*     END
*                      DIMENSION XF(1),NT(20)
*                      DIMENSION MAXXJ(20,2)
*                      INTEGER X,XF
*                      WRITE(6,100) ID,JT
*     100              FORMAT(1H1,10X,'THE FOLLOWING ARE THE FINAL STATE VECTORS
*              . DETERMINED FOR PROGRAM ',A6,' WHICH MET FINAL CONDITIONS AT'/11X,
*              .       'TIME =',I7//5X,'LINE',5X,'COST')
*                      WRITE(6,102) (NN,NN=1,NX)
*     102              FORMAT(1H+,20X,10(9X,'X',I2)/)
*                      DO 10  I=1,NJ
*                      K=I+MAXXJ(JTTT,2)
*                      II=XF(I)
*                      XV=BOOL(X(II,2))
*                      CALL UNPCKS(X,II,NT,NX)
*                      WRITE(6,101) K,XV,(NT(J),J=1,NX)
*     101              FORMAT(1H0,I8,F10.3,10(4X,I8))
*     10               CONTINUE
*                      RETURN
*                      ENTRY PRTOUF(X,JT,XF,NJ,NX,ID)
*                      WRITE(6,100) ID,JT
*                      WRITE(6,102) (NN,NN=1,NX)
*                      DO 20  I=1,NJ
```

```
*                   II=XF(I)
*                   XV=BOOL(X(II,2))
*                   CALL UNPCKS(X,II,NT,NX)
*                   WRITE(6,101) II,XV,(NT(J),J=1,NX)
*     20            CONTINUE
*                   RETURN
*                   END

OF COMPILATION:        NO  DIAGNOSTICS.
```

```
    GPDFIL.PRTCHN,PRTCHN
OMPILED BY 1201 BCS7E ON 18 JAN 71 AT 10:47:17.

NE PRTCHN    ENTRY POINT 000464

USED: CODE(1) 000536; DATA(0) 000223; BLANK COMMON(2) 000000

 REFERENCES (BLOCK, NAME)

NCOD$
UNPCKS
UNPCKP
NIO1$
NIO2$
NWDU$
NREW$
NRBU$
NERR3$

ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

000261 10L       0000   000146 100F     0000   000117 1000F     0000   000130 1001F
000052 127G      0001   000111 142G     0001   000131 152G      0001   000216 203G
000243 220G      0001   000247 224G     0001   000307 236G      0001   000325 246G
000411 277G      0001   000152 3L       0001   000347 30L       0001   000426 307G
000445 60L       0001   000233 61L      0000 I 000000 CP        0000 I 000024 CS
000050 FMT1      0000 I 000063 FMT2     0000 I 000101 I         0000 I 000110 ID
000115 J         0000 I 000100 JJ       0000 I 000106 JTN       0000 I 000103 K
000104 LINPG     0000 I 000077 LPST2    0000 I 000112 LSKIP     0000 I 000116 M1
000107 NSKIP     0000 R 000113 SKIP     0000 R 000111 SPECS     0000 R 000102 XV


*                    SUBROUTINE PRTCHN(X,JT,XFIN,NFIN,NX,NCONTR,NPAGE,LENGP)
*                    INCLUDE DECL,LIST
*                      PARAMETER MAXX=2500
*                      DIMENSION X(MAXX,20)
*     END
*                    DIMENSION CP(20),CS(20),XFIN(1) ,LENGP(1)
*                    DIMENSION FMT1(11),FMT2(11)
*                    INTEGER CURPG
*                    INTEGER CS,CP
*                    INTEGER X,XFIN,FMT1,FMT2
*     C
*     C       SET UP FORMATS FOR PRINTOUT.
*     C
*                    ENCODE(FMT1,1000) NX
*      1000          FORMAT('(1H0,1X,4HTIME,4X,4HCOST,7X,',I2,'(2X,1HX,I2))')
*                    ENCODE(FMT2,1001) NX
*      1001          FORMAT('(1H0,1X,I4,F8.1,7H  STATE,',I2,'(2X,I3))')
*      1002          FORMAT(1H0,2X,'CONTROL',11X,20(2X,I3))
*                    CURPG=NPAGE
```

```
*                 LPST2=(NX-1)/4+(NCONTR-1)/4+4
*                 DO 50  JJ=1,NFIN
*     C
*     C      PICK UP EACH FINAL VECTOR SPECIFIED IN XFIN
*     C
*                 I=XFIN(JJ)
*                 CALL UNPCKS(X,I,CS,NX)
*     C
*     C      WRITE OUT TITLES, LABELS, AND FINAL VECTOR.
*     C
*                 WRITE(6,100) JT
*      100        FORMAT(1H1,10X,'BEGIN BACKCHAIN AT TIME = ',I4,' WITH SEL
*          .ECTED FINAL VECTOR')
*                 XV=BOOL(X(I,2))
*                 WRITE(6,FMT1) (K,K=1,NX)
*                 WRITE(6,FMT2) JT,XV,(CS(K),K=1,NX)
*     C
*     C      RETRIEVE PAGE OF LINK
*     C      RETRIEVE LINK TO PREVIOUS STATE
*     C      UNPACK THE PERMUTATION THAT GENERATED CURRENT STATE
*     C
*                 LINPG=FLD(0,18,X(I,1))
*                 LINLIN=FLD(18,18,X(I,1))
*                 CALL UNPCKP(X,I,CP,NCONTR,NX)
*                 JTN=JT
*     C
*     C      CONSIDER PREVIOUS TIME
*     C
*      3          JTN=JTN-1
*                 IF (JTN .LT. 0)  GO TO 30
*     C
*     C      IF EQUAL, STIL WITHIN SAME PAGE
*     C
*                 IF (LINPG .EQ. CURPG) GO TO 10
*     C
*     C      MUST RETRIEVE PREVIOUS PAGE
*     C
*                 REWIND 10
*                 NSKIP=LINPG-1
*                 READ(10) ID
*                 READ(10) SPECS
*                 IF (NSKIP .EQ. 0)  GO TO 61
*                 LSKIP=0
*                 DO 5  K=1,NSKIP
*      5          LSKIP=LENGP(K)+LSKIP
*                 DO 6  K=1,LSKIP
*      6          READ(10) SKIP
*      61         NLEN=LENGP(LINPG)
*                 DO 7  K=1,NLEN
*      7          READ(10) (X(K,J),J=1,LPST2)
*                 CURPG=LINPG
*     C
*     C      UNPACK STATE AND WRITE OUT AS PREVIOUS STEP IN CHAIN
*     C
*      10         CALL UNPCKS(X,LINLIN,CS,NX)
*                 XV=BOOL(X(LINLIN,2))
*                 WRITE(6,1002) (CP(K),K=1,NCONTR )
```

```
*                   WRITE(6,FMT2) JTN,XV,(CS(K),K=1,NX)
*     C       RETRIEVE PAGE OF LINK
*     C       RETRIEVE LINK TO PREVIOUS PAGE
*     C       UNPACK PERMUTATION THAT GENERATED CURRENT STATE
*     C
*                   CALL UNPCKP(X,LINLIN,CP,NCONTR,NX)
*                   LINPG=FLD(0,18,X(LINLIN,1))
*                   LINLIN=FLD(18,18,X(LINLIN,1))
*                   GO TO 3
*     C
*     C       IF NOT ON LAST VECTOR FOR BACKCHAIN, REINITIALIZE FOR NEXT BACK-
*     C       CHAIN PROBLEM
*     C
*      30           IF (JJ .EQ. NFIN) GO TO 60
*                   CURPG=NPAGE
*                   M1=CURPG-1
*                   LSKIP=0
*                   DO 35  K=1,M1
*      35           LSKIP=LENGP(K)+LSKIP
*                   READ(10) ID
*                   READ(10) SPECS
*                   DO 36  K=1,LSKIP
*      36           READ(10) SKIP
*                   NLEN=LENGP(CURPG)
*                   DO 37  K=1,NLEN
*      37           READ(10) (X(K,J),J=1,LPST2)
*      50           CONTINUE
*      60           RETURN
*                   END
OF COMPILATION:          NO  DIAGNOSTICS.
```

```
      FINVEC,FINVEC
L COMPILED BY 1201 BCS7E ON 07 JAN 71 AT 14:15:40.

JIINE FINVEC    ENTRY POINT 000054

GE USED: CODE(1) 000073; DATA(0) 000030; BLANK COMMON(2) 000000

ON BLOCKS:

   PARAM  000002

NAL REFERENCES (BLOCK, NAME)

    UNPCKS
    NERR3$

AGE ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

L   000015 116G       0000 I 000000 CS        0000 I 000013 I         0000   000015 I
3   000000 JTIME      0000 I 000012 NFJ       0003 I 000001 XIN


   1*                     SUBROUTINE FINVEC(X,JT,MAXXJ,NX,XFIN,NJ)
   2*                     DIMENSION MAXXJ(20,2),XFIN(1),CS(10)
   3*                     INCLUDE DECL,LIST
   3*                       PARAMETER MAXX=2500
   3*                       DIMENSION X(MAXX,20)
   3*      END
   4*                     INCLUDE COMLNK,LIST
   4*                       COMMON/PARAM/ JTIME,XIN(1)
   4*                       INTEGER XIN
   4*      END
   5*                     INTEGER X,XFIN,CS
   6*     C
   7*     C         VECTORS FOR THIS TIME EXTEND FROM MAXXJ(JT+1),1) TO MAXXJ(JT+1,2)
   8*                     NJ=0
   9*                     NFJ=MAXXJ(JT+1,2)-MAXXJ(JT+1,1)+1
  10*                     DO 1000  I=1,NFJ
  11*                     JJ=MAXXJ(JT+1,1)+I-1
  12*                     CALL UNPCKS(X,JJ,CS,NX)
  13*                     INCLUDE FINCNS,LIST
  13*      END
  14*                     NJ=NJ+1
  15*                     XFIN(NJ)=JJ
  16*      1000           CONTINUE
  17*                     RETURN
  18*                     END

END OF COMPILATION:        NO  DIAGNOSTICS.
```

```
       CONSTR,CONSTR
) COMPILED BY 1201 BCS7E ON 07 JAN 71 AT 14:13:55.

JTINE CONSTR    ENTRY POINT 000016

GE USED: CODE(1) 000022; DATA(0) 000005; BLANK COMMON(2) 000000

ON BLOCKS:

   PARAM  000002

NAL REFERENCES (BLOCK, NAME)

   NERR4$
   NERR3$

GE ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

 I  000000 INJP$     0003   000000 JTIME     0003 I 000001 XIN



*DIAGNOSTIC*  THE NAME ST APPEARS IN A DIMENSION OR TYPE STATEMENT BUT IS NEVER REFEI
   1*                  SUBROUTINE CONSTR(CS,CP,NX,NC,$)
   2*                  DIMENSION CS(1),CP(1)
   3*                  INTEGER ST,CP,CS
   4*    C
   5*    C        SUBROUTINE CONSTR TESTS THE FEASIBILITY OF THE STATE AND CONTROL
   6*    C        CONFIGURATION AT TIME JTIME.  THREE TYP3\ OF CONSTRAINTS MUST
   7*    C        BE CHECKED - STATE, CONTROL, AND COMBIN?TION OF STATE AND CONTROL
   8*    C
   9*    C        THREE PROCEDURES PROVIDED BY THE USER D3 THIS - SCONST,CCONST,BCONST
  10*    C
  11*                  INCLUDE COMLNK,LIST
  11*                    COMMON/PARAM/ JTIME,XIN(1)
  11*                    INTEGER XIN
  11*     END
  12*                  INCLUDE SCONST,LIST
  12*     END
  13*                  INCLUDE CCONST,LIST
  13*     END
  14*                  INCLUDE BCONST,LIST
  14*     END
  15*                  RETURN
*DIAGNOSTIC*  CONTROL CAN NEVER REACH THE NEXT STATEMENT
  16*     1000         RETURN 5
  17*                  END

END OF COMPILATION:          2 DIAGNOSTICS.
```

```
       COMCOS,COMCOS
0 COMPILED BY 1201 BCS7E ON 07 JAN 71 AT 14:13:05.

UTINE COMCOS    ENTRY POINT 000016

GE USED: CODE(1) 000022; DATA(0) 000005; BLANK COMMON(2) 000000

ON BLOCKS:

,   PARAM  000002

INAL REFERENCES (BLOCK, NAME)

;   NERR4$
)   NERR3$

IGE ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

)   000000 INJP$      0003   000000 JTIME      0003 I 000001 XIN


    1*                      SUBROUTINE COMCOS(CS,CP,ST,P,V,VV,$)
    2*                      DIMENSION CS(1),CP(1),ST(1)
    3*                      INTEGER CS,CP,ST
    4*     C
    5*     C        COMCOS COMPUTES COST VV FROM THE STATE AND CONTROL AND PARTIAL
    6*     C        COST.  IT ALSO CHECKS IF THE COST SATISFIES ANY CONSTRAINTS.
    7*     C        PROCEDURES TCOST AND VCONST ARE USED.
    8*     C
    9*                      INCLUDE COMLNK,LIST
    9*                        COMMON/PARAM/ JTIME,XIN(1)
    9*                        INTEGER XIN
    9*      END
   10*                      INCLUDE TCOST,LIST
   10*      END
   11*                      INCLUDE VCONST,LIST
   11*      END
   12*                      RETURN
*DIAGNOSTIC*  CONTROL CAN NEVER REACH THE NEXT STATEMENT
   13*      1000            RETURN 7
   14*                      END

END OF COMPILATION:          1 DIAGNOSTICS.
```

```
       COSLIM,COSLIM
 0 COMPILED BY 1201 BCS7E ON 07 JAN 71 AT 14:12:06.

 UTINE COSLIM     ENTRY POINT 000016

 GE USED: CODE(1) 000022; DATA(0) 000005; BLANK COMMON(2) 000000

 ON BLOCKS:

 ,   PARAM  000002

:NAL REFERENCES (BLOCK, NAME)

 ,   NERR4$
 ;   NERR3$

\GE ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

 )   000000 INJP$      0003   000000 JTIME     0003 I 000001 XIN


   1*                      SUBROUTINE COSLIM(P,$)
   2*    C
   3*    C         COSLIM TESTS LIMIT ON COST P AT ANY STAGE IN CALCULATIONS.
   4*    C         CONSTRAINTS COME FROM PROCEDURE COSTCN
   5*    C
   6*                      INCLUDE COMLNK,LIST
   6*                        COMMON/PARAM/ JTIME,XIN(1)
   6*                        INTEGER XIN
   6*     END
   7*                      INCLUDE COSTCN,LIST
   7*     END
   8*                      RETURN
*DIAGNOSTIC*  CONTROL CAN NEVER REACH THE NEXT STATEMENT
   9*     1000             RETURN 2
  10*                      END

END OF COMPILATION:         1 DIAGNOSTICS.
```

```
     COST,COST
 0 COMPILED BY 1201 BCS7E ON 07 JAN 71 AT 14:11:49.

 ION COST          ENTRY POINT 000011

 GE USED: CODE(1) 000013; DATA(0) 000007; BLANK COMMON(2) 000000

 ON BLOCKS:

 )    PARAM   000002

 NAL REFERENCES (BLOCK, NAME)

 F    NERR3$

 AGE ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

 J R 000000 COST       0000    000002 INJP$      0003    000000 JTIME      0000 R 000001 P



*DIAGNOSTIC*  THE VARIABLE, P, IS REFERENCED IN THIS PROGRAM, BUT IS NOWHERE ASSIGNE
*DIAGNOSTIC*  THE NAME ST APPEARS IN A DIMENSION OR TYPE STATEMENT BUT IS NEVER REFE
   1*                   FUNCTION COST(CS,CP,NX,NC)
   2*                   DIMENSION CS(1),CP(1)
   3*                   INTEGER CS,CP,ST
   4*     C
   5*     C       COST COMPUTES PRICE P OF CONTROLS CP APPLIED TO STATE CS.
   6*     C       FUNCTION COMES FROM PROCEDURE COSTF
   7*     C
   8*                   INCLUDE COMLNK,LIST
   8*                     COMMON/PARAM/ JTIME,XIN(1)
   8*                     INTEGER XIN
   8*      END
   9*                   INCLUDE COSTF,LIST
   9*      END
  10*                   COST=P
  11*                   RETURN
  12*                   END

END OF COMPILATION:          2 DIAGNOSTICS.
```

```
       NEXTST,NEXTST
0 COMPILED BY 1201 BCS7E ON 07 JAN 71 AT 14:10:37.

:UTINE NEXTST    ENTRY POINT 000006

,GE USED: CODE(1) 000010; DATA(0) 000005; BLANK COMMON(2) 000000

:ON BLOCKS:

)    PARAM  000002

RNAL REFERENCES (BLOCK, NAME)

+    NERR3$

AGE ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

)    000000 INJP$      0003   000000 JTIME      0003 I 000001 XIN


                      SUBROUTINE NEXTST(CS,CP,ST,NX,NC)
                      DIMENSION CS(1),CP(1),ST(1)
                      INTEGER CS,CP,ST
      C
      C       THIS SUBROUTINE USES A PROBLEM SPECIFIC FUNCTION LOCATED IN
      C       PROCEDURE NEWST TO COMPUTE THE NEW STATE FROM  CS AND CP.
      C
                      INCLUDE COMLNK,LIST
                        COMMON/PARAM/ JTIME,XIN(1)
                        INTEGER XIN
       END
                      INCLUDE NEWST,LIST
       END
                      RETURN
                      END

END OF COMPILATION:        NO  DIAGNOSTICS.
```

Subject: A Dynamic Programming Computer Program
Case 105-4

Author: J. E. Nahra, M. P. Odle

DISTRIBUTION LIST

NASA Headquarters

P. F. Culbertson/MLA
V. Huff/MTE
A. S. Lyman/MA-2
J. W. Wild/MTE

Bellcomm, Inc.

G. M. Anderson
G. C. Bill
A. P. Boysen, Jr.
J. O. Cappellari, Jr.
K. R. Carpenter
D. A. DeGraaf
J. P. Downs
D. R. Hagner
W. G. Heffron
H. A. Helm
J. J. Hibbert
N. W. Hinners
D. P. Ling
H. S. London
K. E. Martersteck
H. H. McAdams
J. Z. Menard
J. M. Nervik
G. T. Orrok
P. F. Sennewald
R. V. Sperry
W. Strack
C. M. Thomas
W. B. Thompson
J. W. Timko
R. L. Wagner
M. P. Wilson
All Members, Center 101
All Members, Center 103
All Members, Center 201
All Members, Department 2032
Department 1024 File
Central Files
Library